**COMPUTER SCIENCE & ENGINEERING**

**Paper – I OOPS & JAVA**

**1. Define OOP**

A) Object oriented programming is methodology or Para diagram to design a program using classes and objects. It simplifies the software development and maintains by providing same concepts.

Object

Class

Inheritance

Abstraction

Encapsulation.

**2. Who invented Java?**

A) “James Gosling” “Mike Sheridan” and Patrick “Naughton” initiated the Java Language Project in June 1991. The small team of Sun Engineers called “Green Team”. Java is a general – purpose object – oriented programming language developed by “Sun Micro Systems” of U.S.A. in 1991. Originally it is called OAK by James.

**3. Need of object oriented program?**

**A**) Object oriented programming allows us to decompose a problem into a number of entities called object and then build data and functions around these entities.

it binds data more closely to the functions that operate on it and protects it from outside.

**4. What is class?**

**A)** Collection of objects is called class. It is a logical entity. A class can also be defined as a blueprint from which you can create an individual object class doesn’t consume any space.

**5. What is object?**

**A)** An entity that has state and behaviour is known as an object. it can be physical or logical. An object can be defined as an instance of a class. An object contains an address and takes up some space in memory.

**6. What is data abstraction?**

**A)** Abstraction is the process of where you show only “relevant” data and “hide” unnecessary details of an object from the user.

**7. Expand JVM, JRE, JIT, OOP**

**A)** JVM : Java Virtual Machine

JRE : Java Runtime Environment.

JIT : Just in Time Compiler

OOP : Object Oriented Programming.

**8. What is JVM?**

**A)** Java virtual machine is virtual machine that can executes Java byte Code. It is the Code execution component of the Java Software platform.

Virtual Machine

Compiler

Program

**Source Program:**

Machine

Intepreter

ByteCode

**9. What is JRE?**

A) Java run time environment is also called Java run time. It is a part of Java development kit (JDK). A set of programming tools for developing Java Application.

**10. What is JIT?**

**A)** In Java programming language and environment a just-in-time. In computing JIT compilation also known as dynamic Translation, it a method to improve to runtime performance of computer programs based on the byte code.

**11. Byte code?**

**A)** Java byte code is the form of instruction of JVM executes. It is intermediate representation of the program. This is machine independent.

**12. Machine neutral?**

**A)** The compiler converts the source code files into byte code files. Those codes are machine – independent and therefore can be run on any machine.

**Long Answer Type Questions**

**13. Explain OOPS Concept?**

**A)** Object oriented programming is methodology or Para diagram to design a program using classes and objects. It simplifies the software development and maintains by providing same concepts.

Object

Class

Inheritance

Abstraction

Encapsulation.

**Object:** An entity that has state and behaviour is known as an object. it can be physical or logical. An object can be defined as an instance of a class. An object contains an address and takes up some space in memory.

**Class:** Collection of objects is called class. It is a logical entity. A class can also be defined as a blueprint from which you can create an individual object class doesn’t consume any space.

**Abstraction**: Abstraction is the process of where you show only “relevant” data and “hide” unnecessary details of an object from the user.

**Inheritance** : when one object acquires all the properties and behaviour of parent object. It is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

**Encapsulation** : Encapsulation is a programming mechanism that binds together code and the data it manipulates, and that keeps both safe from outside interference and misuse. In an object-oriented language, code and data can be bound together in such a way that a self-contained black box is created.

**Polymorphism:** If one task is performed by different ways. It is known as polymorphism. In java example we use method overloading and method overloading to achieve polymorphism.

**14. Explain the features of Java?**

**A)** Java programs can be easily moved from one computer system to another anywhere and anytime changes and upgrades in the system

**Simple:** Java is a small and simple language many feature of c and c++ that are either redundant or source of unreliable code are not part of java. Java uses many constructs of c and c++.

**Object:** Java is true object oriented language. Almost everything in java is an object. All program code and data reside within objects and classes.

**Robust and Security**: Java is robust language. It provides many safe guards to ensure reliable code. It has strict compile time and runtime checking for data types. It maintains garbage collection.

**Distributed**: Java is designed as distributed for creating applications on networks. It has ability to share both data and programs. Java applications can open and access remote objects an Internet as easily as they can do in a local system.

**Multithreading**: Multithreading means handling multiple tasks simultaneously. This means that we need not wait for the application to finish on task before beginning another.

**Dynamic Binding**: Java is a dynamic language. It is capable of dynamically linking is new class libraries, methods and objects.

**Platform Independent:** Java Programs can be easily moved from one computer system to another, anywhere and anytime. Java compiler generates byte code instructions that can be implemented on any machine.

**14. Explain the java program, structure?**

**A) Java Programming Structure:** A Java program may contain many classes of which only one class defines a main method.

To write Java program may contain one or more sections as shown below.

Documentation Section.

Package Statement.

Import Statements.

Interface Statements.

Class definitions

Main method class.

{

Main method definition

}

**Documentation Section**: This section contains set of comment lines giving the name for the program, the author name, software and hardware requirements.

**Package statements:** The first statement allowed in Java program is a package statement.

**Import Statement**: This is similar to the include statement in C. we can decline number of import statements in the program.

**Interface Statements**: An interface is like a class but includes a group of method declarations. This is also optional section.

**Class Definition**: Java program may have multiple system definitions. Classes are the primary and essential element of a java program.

**Main method class**: Every java stand-alone program requires a main method as its starting point; this class is the essential part of a java program. The main method creates objects of various classes and establishes communication between them.

**15. Explain implementation a java program.**

A) **Implementing a Java Program :**

Implementation of a java program involves a series of steps are : create the program

compiling the program

Running the program

Creating the program :

We can create a program using any Text editor. The notepad is very convenient and small text editor. Enter the following program.

Class sample

{

Public static void main (String args[])

{

System.out.println(“Vocational Courses”);

}

}

We must save this program is a file as “Sample.java”.

**Compiling the program** : To compile the program, we must run the java compiler “javac” with the name of the source file on the command line, like

javac sample.java.

**Running the program**: We need to use the java interpreter to run a stand – alone program. At the command prompt type:

Java sample

**15. Explain the web browsers used to Java.**

**A) Web Browsers:** Web browsers are used to pass through the information found on the net.

Example of Browsers:

Hot Java

Chrome

Opera

Mozilla Firefox

Internet Explorer

**Hot Java** : It is the web browser from Sun Micro System that enables to the display of interactive content on the using the java language.

**Chrome Opera** :Opera is a web browser and internet suite developed by opera software. It began in 1994.

**Mozilla Firefox** : It is another popular browser developed by Micro soft for Windows-95, NT and XP work stations.

**Netscape navigator** : This web browser from Netscape communication corporation is a general – purpose browser that can run java applet. It is available for Windows -95, NT, XP, Solar and apple Macinthosh. It is one of the mostly used browser.

**16. How to install JDK software and set java home on window system.**

**A)** **Installation of JDK** :

To install the JDK software and set JAVA – HOME on a windows system. Install the JDK software

a) Goto http ://java.sun.com/javase/downloads/index.jsp:

b) Select the appropriate JDK software and click download. The JDK software is installed on your computer for example at \programfiles\files\ java\jdk 1.6.0-02. You can move the JDK software to another location if desired.

**To set Java-Home** :

Right click my computer and select properties. On the advanced tab, select environment variables and then edit java home to point to where the JDK software is located,

for example. C: \program files\java\jdk1.6. -0.2.

**Configuring Java** : Once the Java is installed, we need to configure it by adding the java path to the environment variable. PATH.

To set the PATH variable to the java directory, we need to perform the following steps.

Right click on the “My Computer” or “Computer” icon

Open

Manage

Map network drive

Disconnect network drive

Create shortcut

Delete

Rename

Properties

Select the properties option from the drop down

Click the “Environment Variables” button to display the Environment variables dialogue box shown on screen.

The environment variable dialog box is divided into two sessions.

User variables

System variables

Under system variable sections, select the path option below the variable column and click the edit button.

To set java director path variable, append to directory path in the variable value text box separated by a semicolon (;) shown on the screen.

Click OK to close the Edit System Variable dialogue box

Click Ok the close Environment variable dialogue box.

Click OK the close system properties dialogue box and complete the process of configuring java.

**UNIT – II**

**THE JAVA PROGRAMMING LANGUAGE**

**1.Define Variable?**

**A)** A variable is a data name that may be used to store a data value. During execution of a program the values are change is called variable.

Example : Avg (Average)

Ht (height)

Tot((Total)

Tot \_ stu(Total students)

**2.** **Define Constants?** **Types of Constants?**

**A)**  A constant are fixed values those are not changed during the execution of program Java supports several types of constants those are :

Integer Constants

Real Constants

Single character constants

Backslash character constants

**3. What are literals?**

**A)** Literals in Java are a sequence of characters (digits, letters and other characters) that represent constant values to be stored in variables. There are five types of literals in java.

Integer literals

Float literals

Character Literals

String Literals

**4. What are Identifiers?**

**A)** These are programmer –designed tokens. They are used for naming classes, methods, variables, objects, labels, packages and interfaces in a program. Java identifiers follow some rules.

a. They can have alphabet, digits and to underscore.

b. They must not begin with digits.

c. Uppercase and lower case letters are distinct.

d. They can be of any length.

**5. Mixed mode?**

**A)** When one of operands is real and the other is integer , the expression is called mixed-mode arithmetic expression.

**Example:**

(int) 12.0/5 gives result 2. Here 12.0 is float operand , 5 is an integer operand .

**6.** **Casting** **values?**

**A)** The process of converting one data type value into another data type is called casting. It is done is two ways.

Implicit type casting or Automatic conversion.

Explicit Type casting.

**7. Access specifies ?**

**A)** An accesser specifier is a keyword that specifies has to access the members of a class or a class itself. We can use access specifiers before a class and its members. There are four access specifiers available in java.

Private

Public

Protected

Private Protected

Default

**8. What is constructor?**

**A)** It’s special method is used to initialige the instance variables at the time of object creation. Constructor is called and executed only once per object. Constructor are not returning any value, not even void also. **Example :**

Student S1 = new student (); Student S2 = new student (“Murthy”, “Lecturer”);

**9. Define method?**

**A)** Think of a method as a subprogram that acts on data and often returns a value. Each method has its own name.

Example : Public static void main (string [] args)

{

Statement ;

Method 1 ()

Statement ;

Method 2 ()

Statement;

}

You are using methods when you use system.out.println() or system.out.print() they are two basic types of methods.

**10. Define Class?**

**A)** A class is a group of objects that gave common property. It is a template or blueprint from which objects are created. A class in java can contain

• Data member

• Method

• Constructor

• Block

syntax of a class

Class Name [ Extends Super class name]

{

Declarations of members

Method declarations

}

**11. Define Object?**

**A)** A runtime entity that has state and behavior is known as an object. For example a pen , a car, a man etc are the objects. It can be tangible or intangible(physical or logical). An object has three characteristics.

• **State** : Represents the data of an object

• **Behavior** : Represents the behavior of an object

• **Identity** : Identity is typically implemented via a unique ID

**12. Void?**

**A)** Is the return type of the method, it means it doesn’t return any value. The void key word allows us to create methods which do not return a value.

**13. Instance variable?**

**A)** It is used in java by object to store their states. Variables which are defined with the static key word and are outside any method declaration are object specific and are known as instance variable.

**14. this?**

**A)** “this” key word refers to the object of the class where it is used. In other words “this” refers to the object to the object of the present class.

**Long Answer Type Questions**

**13.** **Explain Access class method is java?**

**A)** Object variables and methods are not possible to access from outside of the class. It means we cannot access the properties of the object directly. To do this, we must use concerned object and **dot operator** use.

Example : Object name. variable name = values; Object name. method name (parameter list);

Example :

Rectangle re1 = new rectangle();

Rectangle re2 = new rectangle ();

Re1. Length = 20;

Re1. Width = 30;

Re2. Length = 25;

Re2.width = 40;

No two objects re1 and re2 have different values in their fields.

Re 1 Re2

Re1.length 20 25 Re2.length

Re1.width 30 40 Re2.width

The method get data() can be used to do this work. We can call the get data() method on any Rectangle object to set the values of both length and width.That is :

Rectangle re1 = new Rectangle();

Re1.getdata (20,30)

Example : Class Rectangle

{

Int length, width;

Void get data (int x, int y);

{

Length = x;

Width = y;

}

Void rect area ()

{

Int area = length \* width;

}

}

Class rect area

{

Public static void main (string args[])

{

Int area1, area2;

Rectangle re1 = new rectangle[];

Re 1.length =20

Re 1. Width = 30

Area1 = re1.length \* re1.width;

Re2.getdata (25,40);

Area2 = re2.rectarea();

System.out.println(“area1=”, area1);

System.ouot.println(“area2=”; area2);

}

}

**14. Explain the access specifier in java?**

**A)** An accesser specifier is a keyword that specifies has to access the members of a class or a class itself. We can use access specifiers before a class and its members. There are four access specifiers available in java.

Private

Public

Protected

Private Protected

Default

**Private :** Private members of a class are not accessible any where outside the class. They are accessibly only within the class by the methods of that class.

**Public :** Public members or a class are accessible every where outside the class. So any other program can read then and use them.

**Protected :** Protected members of a class are accessible all classes in the same package and derived classes of this class.

**Private protected :** This type of field visible in all subclasses regardless of what pakage they are in, remember. These fields are not accessible by other classes in the same pakage.

**Default :** If no access specifier is written by programmer, then the java compiler uses a default access specifier. Default members are accessible outside the class, but with same package.

**15.** **Explain the operators in java?**

**A)** Java provides a rich set of operators to manipulate variables we can devide all the java operators in to the following groups.

a. ARITHMETICAL OPERATORS.

b. RELATIONAL OPERATORS.

c. LOGICAL OPERATORS.

d. BIT WISE OPERATORS.

e. ASIGNMENT OPERATORS.

f. INCREMENT AND DECREMENT OPERATORS.

g. CONDITIONAL OPRATORS.

**A. ARITHMATICAL OPERATORS:-** Arithematical operators are used in mathematical expressions in the same way that they are used algebra. The following table its arithematical operators . a=5,b=3;

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Purpose | Arithmetic Expression | Result |
| + | Addition | a+b | 8 |
| - | Subtraction | a-b | 2 |
| \* | Multiplication | a\*b | 15 |
| / | Division | a / b | 1 |
| % | Remainder after integer division | a%b | 2 |

**B.RELATIONAL OPERATORS:-**

There are six following relational operation supported by java language.

OPERATORS PURPOSE

= It is equal to

!= It is not equal to

< It is less than

> Greater than

>= Greater than equal to

<= Less than equal to

These returns result in the form of ‘true’ or ‘false’.

**C.LOGICALOPERATORS:-**

The logical operators are &&(and) , ||(or) and !(not).

|  |  |
| --- | --- |
| **OPERATOR** | **PURPOSE** |
| AND  OR  NOT | &&  ||  ! |

**D.ASSIGNMENT OPERATORS:**

There are following assignment supported by java language.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Expression** | **Equivalent** |
| += | x+=a | x=x+a |
| -= | x-=a | x=x-a |
| \*= | x\*=a | x=x\*a |
| /= | x/=a | x=x/a |
| %= | x%=a | x=x %a |
| <<= | x<<=2 | x=x << 2 |
| >>= | x>>2 | x=x >> 2 |
| &= | x&=2 | X = x & 2 |
| ^= | x ^= 2 | x = x ^ 2 |
| |= | x |= 2 | x = x | 2 |

**E. INCREMENT AND DECREMENT OPERATORS :-** They are two increment and decrement operators ++ and - - . These two operators are unique in that can be written both before the operand there are applied to called prefix increment and decrement (+ + i / - - i) or often called prefix increment, decrement(i + + or i - -) the meaning is different in each case.

**F.CONDITIONAL OPERATORS :-**Conditional operators is also known as that ternary operators . This operator consists of three operands and is used to evaluate.

SYNTAX:- variable x =(conditional expression) ? expression1: expression2

**16. Explain logical operators in java?**

**A)** The logical operators are &&(and) , ||(or) and !(not).

|  |  |
| --- | --- |
| **OPERATOR** | **PURPOSE** |
| AND  OR  NOT | &&  ||  ! |

**&& (and) :**

If both operands are true then the logical &&(and) is true

|  |  |  |
| --- | --- | --- |
| a | B | a && b |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**|| (or) :** If either operand is true then the logical ||(or) is true.

|  |  |  |
| --- | --- | --- |
| a | b | a || b |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**! (not) :** It negates the operand.

|  |  |
| --- | --- |
| a | !a |
| 0 | 1 |
| 1 | 0 |

**Ex:** a, b and c are integer variables and assigned 3, 5 and 10 respectively.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Purpose** | **Expression** | **Result** |
| && | And | (a>b)&&(c>a) | False |
| || | Or | (a>b)|| (c>a) | True |
| ! | Not | !(a>8) | True |

**17. Explain Relational Operators with examples?**

**A)Relational Operators: Relational Operators:** There are six relational operators supported by Java language. These returns result in the form of ‘true’ or ‘false’.

**Ex**: a, b and c are integer variables and assigned 3, 5 and 10 respectively.

There are six following relational operation supported by java language.

OPERATORS PURPOSE

= It is equal to

!= It is not equal to

< It is less than

> Greater than

>= Greater than equal to

<= Less than equal to

These returns result in the form of ‘true’ or ‘false’.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Purpose** | **Relational Expression** | **Result** |
| = = | is Equal to | c = = 10 | False |
| ! = | is Not equal to | a != b | True |
| > | is Greater than | a>b | False |
| < | is Less than | (a+b)<c | True |
| > = | is Greater than or equal to | a>=3 | True |
| < = | is Less than or equal to | b < = a | False |

**18. Write about separators in Java?**

**A)**

|  |
| --- |
|  |
| **SEPARATORS** |  | **PURPOSE** |
| Parentheses ( ) |  | Encloses arguments in method definitions and calling; adjusts precedence in arithmetic expressions; surrounds cast types and delimits test expressions in flow control statements |
|  |  |  |
| Braces { } |  | defines blocks of code and automatically initializes arrays |
| Brackets [ ] |  | declares array types and dereferences array values |
| Semi colon ; |  | terminates statements |
| Comma , |  | separates successive identifiers in variable declarations; chains statements in the test, expression of a for loop |
| Period . |  | Selects a field or method from an object; separates package names from sub-package and class names |
| : |  | Used after loop labels |

**19. Explain Bitwise and Bitshift operators in Java?**

**A) BITWISE and BIT SHIFT OPERATORS:** bitwise and bit shift operators are used to manipulate the contents of variables at a bit level according to binary format.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Name of the Operator** | **Example** |
| ~ | Unarybitwise complement | ~op2 |
| & | Bitwise AND | op1 & op2 |
| | | Bitwise inclusive OR | op1 | op2 |
| ^ | Bitwise exclusive OR | op1 ^ op2 |
| << | Signed left shift | op1 << op2 |
| >> | Signed right sift | op1 >> op2 |
| >>> | Unsigned right shift | op1 >>> op2 |

**20. Explain about java Data Types?**

**A)** They are two types of data types in java

Primitive data types

Non-primitive data type.

**Primitive data types :** The primitive data types includes Boolean, char byte int, short in long, float and double.

**Non Primitive data types :** These are classes interfaces and Arrays.

**21. Write about Method Overloading?**

**A)** In java it is possible to create methods that have the same name, but different parameter lists and different definitions. This is called method overloading. It is used when objects are required perform similar.

**Rules for overloading:**

Then may be a difference in the number of parameter passed to the

methods..

There may be difference is the data types of parameters.

There may be difference in the sequence of the parameters.

**Example :**

Class room

{

Float length;

Float breadth;

Room (float x, float y)

{

Length = x;

Breadth = y;

}

Room (float y)

{

Length = breadth = y;

}

Int area()

{

Return (length, breadth)

}

}

Here, we an overloading the constructor method room (). Room r1 = new room(25,20) on the hand, if the room is square then we may create the corresponding object as Room r2 = new room (40 ).

**UNIT 3**

**Control statements and Array**

**1 . If statement?**

**A )** if (test \_ condition)

{

statement\_ block;

}

statement \_ X;

In the above simple if syntax, the statement - block may be a single or group of statements. If the test \_ condition is true. The statement – block will be executed, otherwise the statement will not be executed.

**2. Break statement ?**

**A)** The break statement is used to quit from loop. It can be used with in if, switch(), for (), while () and do- while (). When break statement is encountered in loop, automatically the control will come out of the loop.

Syntax :- break;

**3. Continue statement ?**

**A)** The continue statement tells the compiler “SKIP THE FOLLOWING STATEMENTS AND CONTINUE WITH NEXT ITERATION”. It will work just opposite to the break statement.

Syntax :- continue;

**4. Define an array ?**

**A)** “ An array is defined as a set of homogeneous (same datatype) data items. They can have a common name and stoned in continues locations of the memory”.

**Declaration of Array :-** Syntax :- Data type Array – name [size]

**5. Types of arrays?**

**A)** Arrays can be classified in to :

Single dimensional array or Linear array

Double dimensional array, (Non- Linear array)

Multi dimensional array

**6. What is a string array?**

**A)** “String represents a sequence of characters”. In java we can handle these strings by using character array as :

Char a [ ] = new char [6];

A[0] = ‘M’, a[1]=’U’, a[2]=’R’ a[3]=’T’ a[4]=’H’ a[5]=’Y’

i.e. “MURTHY”

**7. Buffer classes?**

**A)** It is a pure class of string, while string creates strings of fixed- length, string buffer creates strings of flexible length that can be modified in terms of flexible length that can be modified in terms of both length and content.

**8. String method?**

**A)** The string class defines a number of methods that allows us to accomplish a variety of string manipulation task.

1) S2 = S1.to lower case (); Convert the string S1 to all lower case.

2)S2=S1. Upper case(); Convert the string S1 to all upper case.

3) S2=S1. Replace (‘a’,’b’) Replace all appearances of a with b

4) S2=S1. Equals ignore case (s2) Returns true if s1 is equal to s2.

5) S1.equals (S2) Returns a true if S1 is true to S2.

**9. Nested loops?**

**A)** A for loop or while loop in another for loop or while loop is called nested loop. A loop in another loop is called nested loop.

**Long Answer Type Questions**

**10. Explain the conditional statement in java?**

**A) Simple if statement:-**

Syntax:- if (test \_ condition)

{

statement\_ block;

}

statement \_ X;

In the above simple if syntax, the statement - block may be a single or group of statements. If the test \_ condition is true. The statement – block will be executed, otherwise the statement will be skipped.

**TEST Condition**

Statement block

False True

Next Statement

Statement x

**If ------ else statement:**

The if – else statement is used to execute true or false.

**Syntax:-**

if (test \_ expression)

{

true \_ block statements;

}

else

{

false\_ block statement;

}

statement \_ X;

If the test\_expression is true , then the true\_block statements are executed true\_ block may have one or more statements if the test\_expression is false (else) the false\_block statements are executed.

**TEST Condition**

False True

FalseStatement block

True Statement block

Next Statement

**.Nested if statement:-** When a series of decision are involved, we may have to use more than one if else statement in nested form.

**Syntax:-**  if ( test \_ condition \_ 1)

{

if (test \_condition \_2)

{

statement \_ 1;

}

else

{

statement \_2;

}

else

{

statement \_ 3;

}

statement \_ n;

**TEST Condition 1**

False True

Statement x

False True

**Condition2**

conditio

Statement 3

Statement 2

Statement 1

**ELSE……IF ladder:**

Another way of using if’s together when multipath are involved. A multipath decision is a chain of ifs in which the statement associated with each else is an if.

**Syntax:-**

if (condition\_1)

statement \_1;

else if ( condition \_2)

statement \_ 2;

else if ( condition \_3)

statement \_ 3;

-----

------

else

default \_statement;

statement\_ X;



**11. Explain switch statement with example?**

**A) Switch Statement:** The switch statement tests the value of a given variable against a list of value and when a match is found, corresponding block of statements associated with the case will be executed. If none is matched ‘default’ block will be executed. The ‘break’ statement at the end of each block signals the end of a particular case causes an exit from the switch statement, transferring the control to the statement immediately following the switch.

**Syntax:** switch(expression)

{

case value: block1;

break;

case value: block2;

break;

case value: block3;

break;

--------------------

--------------------

default: default block;

break;

}

Sample program with switch.. case is as follows.

import java.util.Scanner;

public class SwitchExample

{

public static void main(String args[])

{

System.out.println("enter a code");

Scanner in = new Scanner(System.in);

int code = in.nextInt();

switch(code)

{

case 1 :

System.out.println("first class"); break;

case 2:

System.out.println("second class"); break;

case 3 :

System.out.println("third class"); break;

default :

System.out.println("fail");

}

}

}

**12. Explain the loops in java?**

**A) WHILE STATEMENT:**

**Syntax:-** while (condition)

{

Body of the loop;

}

The while statement in an entry controlled loop statement . The condition is evaluated first and if condition is true, then the body of the loop is executed.

**Example:-**

s = 0;

i = 0;

while (i ≤ 10)

{

s = s+i;

i= i+1;

}

System.out.Println (“sum = 5d \n”, s);

**DO STATEMENT:-**

The do statement is an exit controlled loop statement. The do statement the program proceeds to evaluate the body of loop first. if the condition is true the program continues to evaluate the body of the loop once again.

**Syntax:-**

do

{

Body of the loop;

} While (condition);

**Example:-**

i=0

s=0

do

{

s=s+i;

i=i+1;

} While (I ≤ 10);

**FOR STATEMENT:**

The for is another entry controlled loop statement that provides a more concise loop control structure.

**The syntax of the for loop is:**

For loop is more concise than while loop and do while loop.

for (initial\_value; final\_value; increment value )

{

body of the loop;

}

**Example:-**

S=0;

System.out.println(“enter the value”);

Scanner in =new scanner (system.in);

N=in nextInt();

For(i=0;I,<=n;i++)

{

S=s+i:

}

System.out.println(“sum=”+s);

**13. What are the string method explain?**

**A) String methods :-**

The string class defines a number of methods that allows us to accomplish a variety of string manipulation task.

**String methods :-** The string class defines a number of methods that allows us to accomplish a variety of string manipulation task.

|  |  |
| --- | --- |
| **Method call** | **Task performed** |
| 1) S2 = S1.to lower case (); | Convert the string S1 to all lower case. |
| 2)S2=S1. Upper case(); | Convert the string S1 to all upper case. |
| 3) S2=S1. Replace (‘a’,’b’) | Replace all appearances of a with b |
| 4) S2=S1. Equals ignore case (s2) | Returns true if s1 is equal to s2, ignoring the case of characters. |
| 5) S1.equals (S2) | Returns a true if S1 is true to S2 |
| 6) S2=S1 trim() | Removes white spaces at the beginning and end of the string S1. |
| 7) S1.length () | Gives the length of S1 |
| 8)S1.character | Gives the nth character of S1 |
| 9)S1. Comparator (S2) | Returns negative if (S1S2) and zero if S1 is equal to S2. |
| 10)S1.concat (S2) | Concatenates S1 and S2 |
| 11) S1.substring (n) | Gives substring starting from nth character |
| 12) S1.substring (n, m) | Gives substring starting from nth character up to in the character (not including mth) |
| 13) String.value of (P) | Creates a string object of the parameter P(simple) type of object |
| 14) P. to string () | Creates a string representing of object ‘P’ |
| 15) SI. Index of (“A”) | Gives the first occurrence of ‘A’ in the string SI |
| 16) S1.index of (‘X,n) | Gives the position of ‘X’ that occurs after nth position in the string S1. |
| 17) String value of (variable) | Converts the parameter value to string representation |

**14.** **Explain the string Buffer class in Java?**

**A)** It is a pure class of string, while string creates strings of fixed- length, string buffer creates strings of flexible length that can be modified in terms of flexible length that can be modified in terms of both length and content.

String Buffer append (val) :Val may be Boolean int, short, long etc. it will be added to string buffer object

String buffer sb=new string buffer(“SREE”);

Sd.append (“RAMA”)

String Buffer insert (int I , val)

Val may be any data type, that will be inserted in to the string buffer at the specified position by i.

\*String Buffer delete (int I, int j) : It removes the characters from ith to J-1th position in the string buffer.

\*String Buffer reverse () : It reverse the character sequence in the string buffer

\*int length () : It returns the no of characters in the string buffer

Object.

\*String to string () : This is used to convert string buffer to string

Object.

\*int idex of (string str) : It returns the first occurrence of ‘str’ in

string buffer Object.

\*String Buffer set char at (n,x) : It returns the modified the nth

character to X

\*Int set length (n) : sets the length of the string S1 to n. if nS1.

Length() zeros one added to S1.

**15.** **What are the differences between string buffer and string builder classes?**

**A)**

| **BASIS FOR COMPARISON** | **STRING** | **STRINGBUFFER** |
| --- | --- | --- |
| Basic | The length of the String object is fixed. | The length of the StringBuffer can be increased. |
| Modification | String object is immutable. | StringBuffer object is mutable. |
| Performance | It is slower during concatenation. | It is faster during concatenation. |
| Memory | Consumes more memory. | Consumes less memory. |
| Storage | String constant pool. | Heap Memory. |

Key Differences Between String and StringBuffer

1. The length of String object is fixed but the length of an object of StringBuffer can be increased when required.
2. String object is immutable i.e. it’s object can’t be reassigned again whereas, the object of StringBuffer is mutable.
3. String object is slower in performance whereas, the StringBuffer object is faster.
4. String object consumes more memory whereas, StringBuffer objects consumes less memory.
5. String objects are stored in a constant pool whereas, StringBuffer objects are stored on heap memory.

**16. Write a java program to print 1 to n?**

**A) public** **class** Print\_1\_To\_N\_UsingWhile

{

**public** **static** void main(**String**[] args)

{

//loop counter initialisation

int i =1;

//create object of scanner class

Scanner Sc = **new** Scanner(**System**.in);

// enter the value of " n "

**System**.out.print("Enter the value n : ");

// read the value.

int n = Sc.nextInt();

**System**.out.println("Numbers are : ");

**while**(i<=n)

{

**System**.out.println(i);

i++;

}

}

}

**Output**

Enter the value n : 15

Numbers are :

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

**17.** **Write a java program addition of two matrices?**

**A)**

import java.util.Scanner;

class AddTwoMatrix

{

public static void main(String args[])

{

int m, n, c, d;

Scanner in = new Scanner(System.in);

System.out.println("Enter the number of rows and columns of matrix");

m = in.nextInt();

n = in.nextInt();

int first[][] = new int[m][n];

int second[][] = new int[m][n];

int sum[][] = new int[m][n];

System.out.println("Enter the elements of first matrix");

for ( c = 0 ; c < m ; c++ )

for ( d = 0 ; d < n ; d++ )

first[c][d] = in.nextInt();

System.out.println("Enter the elements of second matrix");

for ( c = 0 ; c < m ; c++ )

for ( d = 0 ; d < n ; d++ )

second[c][d] = in.nextInt();

for ( c = 0 ; c < m ; c++ )

for ( d = 0 ; d < n ; d++ )

sum[c][d] = first[c][d]+ second[c][d];

System.out.println("First Matrix :-");

for ( c = 0 ; c < m ; c++ )

{

for ( d = 0 ; d < n ; d++ )

System.out.print(first[c][d]+"\t");

System.out.println();

}

System.out.println("Second Matrix:-");

for ( c = 0 ; c < m ; c++ )

{

for ( d = 0 ; d < n ; d++ )

System.out.print(second[c][d]+"\t");

System.out.println();

}

System.out.println("Sum of entered matrices:-");

for ( c = 0 ; c < m ; c++ )

{

for ( d = 0 ; d < n ; d++ )

System.out.print(sum[c][d]+"\t");

System.out.println(); } } }

**Output:**

Enter the number of rows and columns of matrix

2

2

Enter the elements of first matrix

1 2

3 4

Enter the elements of second matrix

5 6

7 8

First Matrix :-

1 2

3 4

Second Matrix:-

5 6

7 8

Sum of entered matrices:-

6 8

10 12

**18. Write a java program to sort “N” numbers in an array?**

1. **A)**
2. import java.io.\*;
3. import java.lang.\*;
4. **public** **class** SortAsc {
5. **public** **static** **void** main(String[] args) {
7. //Initialize array
8. **int** [] arr = **new** **int** [] {5, 2, 8, 7, 1};
9. **int** temp = 0;
11. //Displaying elements of original array
12. System.out.println("Elements of original array: ");
13. **for** (**int** i = 0; i < arr.length; i++) {
14. System.out.print(arr[i] + " ");
15. }
17. //Sort the array in ascending order
18. **for** (**int** i = 0; i < arr.length; i++) {
19. **for** (**int** j = i+1; j < arr.length; j++) {
20. **if**(arr[i] > arr[j]) {
21. temp = arr[i];
22. arr[i] = arr[j];
23. arr[j] = temp;
24. }
25. }
26. }
28. System.out.println();
30. //Displaying elements of array after sorting
31. System.out.println("Elements of array sorted in ascending order: ");
32. **for** (**int** i = 0; i < arr.length; i++) {
33. System.out.print(arr[i] + " ");
34. }
35. }
36. }

**Output:**

Elements of original array:

5 2 8 7 1

Elements of array sorted in ascending order:

1 2 5 7 8

**19.Write a java program multiplication of two matrices?**

**A)**

import java.util.Scanner;

class Mulmat{

public static void main(String args[]){

int m,n,i,j,k;

Scanner in=new Scanner(System.in);

System.out.println("Enter the number of rows and columns of matrix:");

m=in.nextInt();

n=in.nextInt();

int a[][]=new int[m][n];

int b[][]=new int[m][n];

int c[][]=new int[m][n];

System.out.println("Enter the elements of first matrix:");

for(i=0;i<m;i++)

for(j=0;j<n;j++)

a[i][j]=in.nextInt();

System.out.println("Enter the elements of b matrix:");

for(i=0;i<m;i++)

for(j=0;j<n;j++)

b[i][j]=in.nextInt();

for(i=0;i<m;i++){

for(j=0;j<n;j++){

c[i][j]=0;

for(k=0;k<2;k++) {

c[i][j]=c[i][j]+(a[i][k]\*b[k][j]); // Multiplication of two matrices

}

}}

for(i=0;i<m;i++){

for(j=0;j<n;j++)

System.out.print(c[i][j]+"\t");

System.out.println();

}}}

**Output:**

Enter the number of rows and columns of matrices 2 2

Enter the elements of first matrix: 1 2 3 4

Enter the elements of second matrix: 1 2 3 4

7 10

15 22

**20. Write a java program to find the factorial of given number?**

1. **A) class** FactorialExample{
2. **public** **static** **void** main(String args[]){
3. **int** i,fact=1;
4. **int** number=5;//It is the number to calculate factorial
5. **for**(i=1;i<=number;i++){
6. fact=fact\*i;
7. }
8. System.out.println("Factorial of "+number+" is: "+fact);
9. }
10. }

**Output:**

Factorial of 5 is: 120

**21.** **Write a java program to find whether the given number is perfect or not?**

**A)**

class Perfect

{

public static void main(String arg[])

{

    long n,sum=0;

                 Scanner sc=new Scanner(System.in);

    System.out.println("Enter a number");

                   n=sc.nextLong();

    int i=1;

    while(i<=n/2)

    {

       if(n%i==0)

       {

sum+=i;

       }

      i++;

    }

if(sum==n)

{

System.out.println(n+" is a perfect number");

               }

else

System.out.println(n+" is not a  perfect number");

}

}

**Output:**

Output

|  |  |
| --- | --- |
| 1  2  3 | Enter a number  6  6 is a perfect number |

**22. Write a java program to transpose of given matrices?**

**A)**

public class Transpose {

public static void main(String[] args) {

int row = 2, column = 3;

int[][] matrix = { {2, 3, 4}, {5, 6, 4} };

// Display current matrix

display(matrix);

// Transpose the matrix

int[][] transpose = new int[column][row];

for(int i = 0; i < row; i++) {

for (int j = 0; j < column; j++) {

transpose[j][i] = matrix[i][j];

}

}

// Display transposed matrix

display(transpose);

}

public static void display(int[][] matrix) {

System.out.println("The matrix is: ");

for(int[] row : matrix) {

for (int column : row) {

System.out.print(column + " ");

}

System.out.println();

}

}

}

**Output:**

The matrix is:

2 3 4

5 6 4

The Resultant matrix is:

2 5

3 6

4 4

**UNIT 4**

**Implementing oops in Java**

**1.What is a method?**

**A)** In object – oriented programming, method is a jargon used for function. Methods are bound to a class and they define the behavior of a class.

**2.** **How to call method?**

**A)** Now you defined a method you need to use it for that, you have to call the method Here’s how :

myMethod (); This statement calls the myMethod() method that was declared earlier.

Class Main

{

Public static void main(String[] args)

{

……………………………..

myFunction();

…… ……. …….

} **(1)**

Private static void myFunction()

{

(3) // function body

……. ……. ……………. (2)

………….. ……….. …..

}

}

**3. What are the advantages of using methods?**

**A)** The main advantages is code reusability. You can write a method once, and use it multiple times. You do not have to rewrite the entire code each time. Think of it as “ Write once, reuse multiple times.”

**4. Constructor?**

**A)** A constructor initializes an object when it is created. It has the same name as its class and is syntactically similar to a method. However, constructors have no explicit return type.

**5. Over loading?**

**A)** When we declare the same method is child class which is already present in the parent class this is called method overloading. In this case when we call the method from child class object, the child class version of the method is called. However we can call the parent class method using super keyword.

**6. Inheritance?**

**A)** The mechanism of deriving a new class from and old one is called inheritance. The old class is known as the base class is sub class or derive class or child class. The inheritance allows sub classes to inherit all the variables and methods of their parent classes. Inheritance may take different forms.

single inheritance (only one super class)

multiple inheritance ( several super classes)

hierarchical inheritance ( one super class many sub classes)

multilevel inheritance (derived from a derived class)

**6.Polymorphism?**

**A)** Polymorphism is one of the oop’s feature that allows us to perform a single action in different ways. For example, we have a class animal that has a method sound (). Since this is a generic class so we can’t give it a implementation like rower, meow, rink etc. we had to give a general message.

**7.** **Overriding?**

**A)** **overriding** is: ability to define a behavior that's specific to the subclass type, which means a subclass can implement a parent class method based on its requirement. In object-oriented terms, **overriding** means to **override** the functionality of an existing method.

**Long Answer Type Questions.**

**8.** **Explain the Java methods with arguments and return value?**

**A)** Public static void main (String [] args)

{

Int result, n;

n=3;

result = square (n);

System.out.println (“Square of 3 is :” + result);

N=4;

Result = square (n);

Syatem.out.println (“Square of 4 is : “+ result);

}

Statc int square (int i)

{

Return I \* I;

}

}

When you run the program, the output will be :

Squared value of 3 is : 9

Squared value of 4 is : 16

Now, the square () method returns the squared value of whatever integer value passed to it.

Class Squaremain

{

Public static void main (String [] args)

{

---- ---- ----

N = 3; (3)

(9) result = square (n);

------- ------

}

Private static square (int i)

{

// return Return i \* i;

(9)

}

}

Java is a strongly – typed language. If you pass any other data type except int ( in the above example ), compiler will throw an error.

The argument passed n to the get square () method during the method call is called actual argument.

Result = get square (n);

The parameter i accepts the passed arguments in the method definition get square (int i). This is called formal argument (parameter). The type of the formal argument must be explicitly typed.

**9.Explain the final variables and methods?**

**A)** All methods and variables can be over ridden by default in subclass. If we want to prevent the subclass from overriding the members of super class. We can declare them as final using the keyword final as a modifier. Methods which are declared as “final” are called final method.

**Ex** : final int size = 100;

Final void show staturs ()

{ …..}

The value of a final variables can never be changer.

**Final classes :** A final is a class which is declared as final find keyword before a class prevents inheritance. This means subclasses cannot created to a final class.

Final class A{ ….}

Final class B extends some class { ……..}

Any attempt to inherit these will cause an error at compilation time.

**Final Methods :** We know constructor method is used to initialize an object when it is created. This process is known as initialization. Similarly java support a concept called finalization. Which is just possible to initialization.

**10.Explain the inheritance with example in java?**

**A) Inheritance:** Inheritance is the process by which objects of one class acquire the properties of objects of another class. Inheritance supports the concept of hierarchical classification. Inheritance provides the idea of reusability.

Through inheritance we can create new classes that are built upon existing Classes. When we inherit from an existing class, we can reuse methods and fields, and we add new methods and fields to adopt our new class to new situations.

Syntax:

class<subclass\_name> extends <superclass\_name>

{

//methods and fields

}

The keyword extends indicates that we are making a new class that derived from an existing class. i.e., a class that is inherited is called superclass. The new class is called subclass.

Ex: class Employee{

int salary=60000;

}

class Programmer extends Employee{

int bonus= 10000;

}

class InterfaceExample{

public static void main(String args[ ]){

Programmer p=new Programmer( );

System.out.println("Programmer salary is: " +p.salary);

System.out.println("Programmer bonus is: " +p.bonus);

}}

Output:

Programmer salary is: 60000

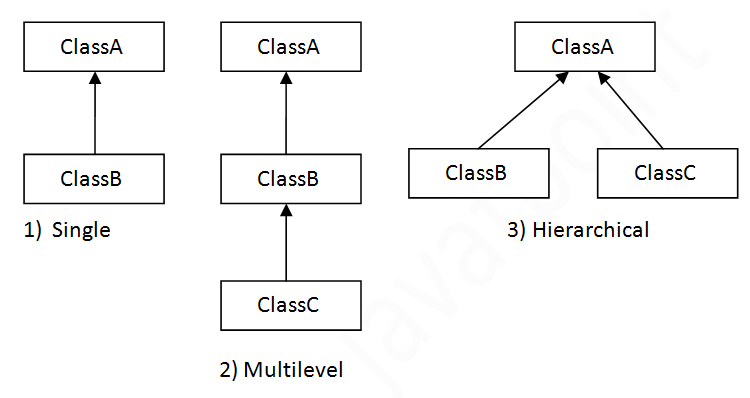
Programmer bonus is: 10000

**11.Types of inheritance in Java?**

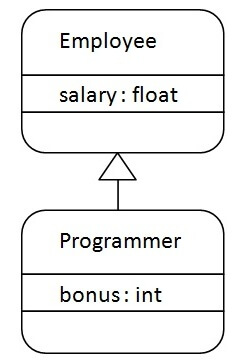
## A) Types of inheritance in java

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

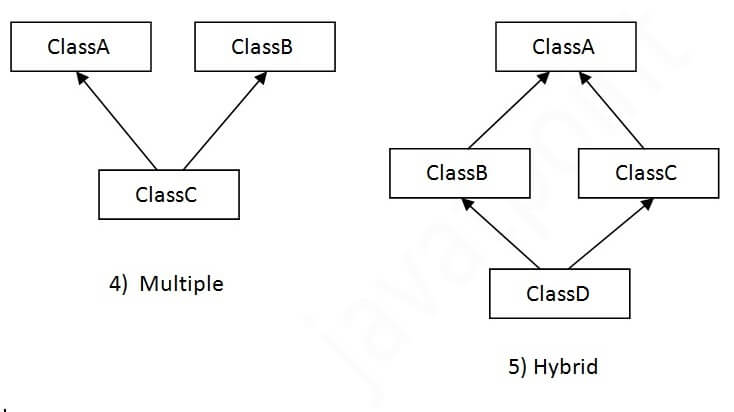
In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.



### **Java Inheritance Example**



When one class inherits multiple classes, it is known as multiple inheritance. For Example:



Below are Various types of inheritance in Java. We will see each one of them one by one with the help of examples and flow diagrams.

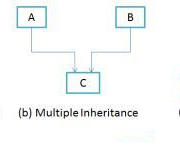
### 1) Single Inheritance

**Single inheritance** is damn easy to understand. When a class extends another one class only then we  call it a single inheritance. The below flow diagram shows that class B extends only one class which is A. Here A is a **parent class** of B and B would be  a **child class** of A.

### [Single Inheritance](https://beginnersbook.com/wp-content/uploads/2013/05/Single-Inheritance.png)

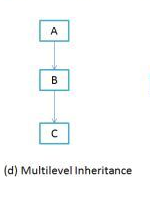
### 2) Multiple Inheritance

“**Multiple Inheritance**” refers to the concept of one class extending (Or inherits) more than one base class. The inheritance we learnt earlier had the concept of one base class or parent. The problem with “multiple inheritance” is that the derived class will have to manage the dependency on two base classes.

[](https://beginnersbook.com/wp-content/uploads/2013/05/Multiple-Inheritance.png)

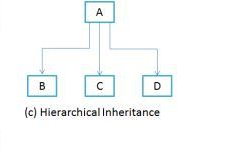
### 3) Multilevel Inheritance

**Multilevel inheritance** refers to a mechanism in OO technology where one can inherit from a derived class, thereby making this derived class the base class for the new class. As you can see in below flow diagram C is subclass or child class of B and B is a child class of A.

[](https://beginnersbook.com/wp-content/uploads/2013/05/Multilevel-Inheritance.png)

### 4) Hierarchical Inheritance

In such kind of inheritance one class is inherited by many**sub classes**. In below example class B,C and D **inherits** the same class A. A is **parent class (or base class)** of B,C & D.

[](https://beginnersbook.com/wp-content/uploads/2013/05/Hierarchical-Inheritance.png)

### 5) Hybrid Inheritance

In simple terms you can say that Hybrid inheritance is a combination of**Single** and **Multiple inheritance.** A typical flow diagram would look like below. A hybrid inheritance can be achieved in the java in a same way as multiple inheritance can be!! Using interfaces. yes you heard it right. By using **interfaces** you can have multiple as well as **hybrid inheritance** in Java.

.

[](https://beginnersbook.com/wp-content/uploads/2013/05/Hybrid-inheritance.png)

**12. Explain the method overloading with example in Java?**

**A)** When we declare the same method is child class which is already present in the parent class this is called method overloading. In this case when we call the method from child class object, the child class version of the method is called. However we can call the parent class method using super keyword.

Example :- Class parent-class

{

Parent class();

{

System.out.println(“construct of parent”);

}

Void disp()

{

System.out.println (“parent method”);

}

}

Class java example extends parentclass

{

Java example ()

{

System.out.println (“Constructor of child”);

}

Void disp()

{

System.out.println (“child method”);

Super disp();

}

Public static void main(string args[])

{

Java example obj = new java example ();

Obj.disp();

}

}

**Out put :-**

Constructor of parent

Constructor of child

Child method

Parent method

**13.** **Explain the operator overloading in Java?**

**A)** Java doesn’t support user defined operators over loading. The only aspect of java which comes close to “custom” operator overloading is the handling of “+” for string, which either result in compile – time concalination of constants or execution – time concatenation using string builder or string buffers which act in the same way through. In addition to all the people pointing out that + is overloaded for string “ -“ over loaded for both floating point and integer operators as are \* and /. % is also overloaded for floating point which can be a bit of surprise for those with a C or C++ back ground.

**Example :-**

Public static void main (String args[])

{

Final string first = “ length: 25;

Final string second = “length + first. Length();

System.out.println (“first and second are equal :” + first == second );

}

It prints false and nothing else .

The + operator, whether used for addition or string concatenation has more precedence than the == operator. Therefore the parameter of the println method is evaluated like. System.out.println (“first and second are equal: “+ first == second);

**14.** **Explain the polymorphism with example?**

**A)** Polymorphism is one of the oop’s feature that allows us to perform a single action in different ways. For example, we have a class animal that has a method sound (). Since this is a generic class so we can’t give it a implementation like rower, meow, rink etc. we had to give a general message.

**Ex:** class Box{

int w,h;

void info( ){

System.out.println("This is a simple box");

System.out.println("width = "+w+"height="+h);

}}

class woodenBox extends Box{

int life;

void info( ){

System.out.println("This is a wooden box");

}}

class SteelBox extends Box{

int wg;

void info( ){

System.out.println("This is a steel box");

}}

class LargewoodenBox extends woodenBox{

void info( ){

System.out.println("This is a Huge wooden Box");

}}

class BoxDemo{

public static void main(String ary[ ]){

Box x;

Box b1= new Box( );

woodenBox wb=new woodenBox( );

SteelBox s1=new SteelBox( );

LargewoodenBox p1=new LargewoodenBox( );

b1.info( );

wb.info( );

s1.info( );

p1.info( );

}}

Output:

This is a simplebox

Width=0 hieght=0

This is a wooden box

This is a steel box

This is a Huge wooden Box

**UNIT 5**

**PACKAGES AND ITERFACES**

**1.Define package?**

**A) Package:** Packages are a way of grouping a variety of classes and or interfaces together. The grouping is done according to their functionality. Packages are act like containers for classes.

**Syntax:**

1. To create a package:

**package package\_name;**

2. To import a package:

**import package\_name.\*;**

**2.** **Create a package?**

**A)** First we declare name of the package using package keyword followed by the package name. This must be first statement in a java source file. Then we define classes for that package.

Ex :- package first package;

Public class first-class

{

------------;

------------;

}

**3.** **Accessing a package?**

**A)** We use the import statement to import the classes packages. The import statement can be used to list of packages for a particular class. The general form of the import statement is

Syntax :- import package 1 [.package 2] [ .package2] [.package3]. class name

**4.** **What is an Interface?**

**A) Interface:** Interface is a kind of class that contains fields and methods which are final fields and Abstract methods. This means, interfaces does not specify any code to implement these methods and data fields contains only constants.

**Defining an Interface:**

**Syntax*:***

interface interface\_name {

Variable Declaration;

Method Declaration;

}

**6.What is Abstract class?**

**A)** An abstract class is written when there are some common feature shared by all the objects. it is declared by using the keyboard “abstract”.

An example abstract class in Java

abstract class Shape {

    int color;

    // An abstract function (like a pure virtual function in C++)

    abstract void draw();

}

**6.Define a sub- class?**

**A)** class subclass name extends superclass name

{

Variables declaration;

Methods declaration;

}

The key word extends signifies that the properties of the super class name are extended to the sub class name. the sub class will now contain its own variables and methods as well those of the super class name.

**Long Answer Type Questions**

**7.Explain the types of packages?**

**A) Types of packages :-** Packages are two types.

Java API packages

Java user defined packages.

**Java API packages** :- They provides a large number of classed grouped in to different packages according to functionality like long. Util , out. Applet etc. each package provides different functionality to our java program. The different packages are :

**Java Long** :- It includes class primitive data types, string math functions.

**Java .util** :- Languages, utility classes such as vector, hash tables, random numbers, date etc.

**Java . Applet** : - It includes classes for windows buttons list menus and abstract window tool kit.

**Java. IO :-** Input output support classes.

**Java . net :-** Net stands for network. Its mainly in client – sever programming.

**Java.text :-** this package is useful to format the numeric values (Date format class for dates) [ number format class for values]

**Java.sql:-** Sql stands for structured query language. This package helps to connect to database like or Sybase, retrieve data from there and used in the java application.

**Java User defined packages:**

While declaring package first declare the name of the package using the packages keyword followed by package name must be the first line in java.

**Syntax** :-

package < package name>;

Example:- package student;

Using one class from other package. The following sample program will be class from other package.

Example :-

package x;

public class y

{

public void display ()

{

System.out.println (“class y”);

}

}

The above example has a package name x and containing a simple class y. this should be name y.java stored in the subsidiary “x”. if we compile this java file, resultant “y” class with stored in the same sub- directory.

**8.** **Explain to create own package with example?**

**A) Steps involved in creating our own package are:**

1. Declare the package at the beginning of a file using the form

**package** <package\_name>;

1. Define the class that is to be put in the package and declare it “public’.
2. Create a subdirectory under the directory where the main source files are stored.
3. Store the listing as the classname.java file in the subdirectory created.
4. Compile the file. This creates “ .class “ file in the subdirectory.

Note: The subdirectory name must match the package name exactly.

**Example:**

package package1;

public class ClassA

{

public void displayA( )

{

System.out.println(“Class A”);

}

}

import package1.ClassA;

class PackageTest1

{

public static void main(String args[ ])

{

ClassA objectA = new ClassA();

objectA.displayA( );

}

}

Package1 containing single class ClassA importing ClassA from package1

**9.** **How to provide implementation to the interface by the implementation a class?**

**A)** In java, we are using the **implements** keyword to implement the methods, which are declared in the corresponding interface.

That is : class< class name > implement <Interface name>

{

Body of the class name

}

An interface is a specification of method prototypes. All the methods of the interface are public and abstract.

Interface methods are public since they should be available to third party vendors to provide implementation.

An interface can have variables, which are public static and final by default. This means all the variables of the interface are constants.

None of methods in interface can be private, protected or static.

if any method is not implemented then ,implementation class should be declared as “abstract”.

An interface can extend another interface.

An interface can not implement another interface.

A class can implements (not extends).multiple interfaces.

**10.** **What are the differences between abstract class and interface?**

**Abstract class vs interface:**

|  |  |
| --- | --- |
| Abstract Class | Interface |
| 1.an abstract class is written when there are some common feature shared by all the objects. | 1.an interface is written when all the features are implemented differently in different objects. |
| 2.when an abstract class is written ,it is duty of the programme to provide sub classes to it. | 2.an interface is written when the programme wants to leave the implementation to the third party vendors . |
| 3.it contains some abstract method and some can create methods | 3.it contains only abstract methods. |
| 4.it is declared by using the keyword abstract | 4.it is declared using the keyboard interface. |
| 5.it also contains instance variables | 5.it contains only constants |
| 6.all the abstract methods of the abstract class should be implemented is its sub classes. | 6.all the methods of the interface should be implemented in its implementation classes. |

**11.** **What are the advantages and disadvantages of interface?**

**A)** **Benefits of packages (or) Benefits by organizing classes into Packages:**

1. Provides reusability of the class by the other programs existing in the same package.
2. In packages, classes can be unique compared with classes in other packages.
3. Packages provide a way to “hide” classes thus preventing other programs or packages from accessing classes that are meant for internal use only.
4. Packages offers a way of separation between “Design” and “Coding” by designing classes and decide their relationships first and implementing the java code needed for the methods next.

**Benefits or Advantages of Interfaces:**

1. The legal implementations for a class are defined in the interface.
2. Java provides an alternate approach for multiple inheritance known as Interfaces to support the multiple inheritance.
3. A java class cannot be subclass of more than one super class but it can implement more than one interface.

**UNIT-6**

**EXCEPTION HANDLING**

**1.** **What is Debugging?**

**A)** A software engineer may also commit several errors while designing the are developing the code. The errors are the wrong that can make a

Program go wrong. These errors are also called “bugs” and the process of removing them is called debugging.

**2.** **Write the Types of errors?**

**A) Errors in java program:** there are basically 3 types of errors in the java program.

Compile Errors

RuntimeErrors

LogicalErrors

**3.** **Define exception?**

**A)**  An exception is a runtime error. It means all exceptions occur only at runtime but some exceptions are detected at compile time and some at runtime.

The exceptions that are checked at compilation time by the java compile are called “checkedexception”.

while the exceptions that are checked by the JVM are called “un checked exceptions”

**4.** **Write about throws clause?**

**A)** The programmer may or may not handle the runtime exceptions In case the programmer does not want to handle ,the checked exceptions , it should throw it out using the ”throws” clause .

**5.** **What is throw clause?**

**A)** simply by using throw class, we can throw the exception object to the corresponding catch block . so throw statements are commonly placed in the try blocks only .

Example :- Throw new IO Exception (“simple”);

This throw clause is used for

**6.** **write difference between error and exception?**

**A) Exception:**

An exception is an error, which can be handled. It means when we exception happens, the programmer can do something to avoid any harm.

**Error:**

An error, which cannot be handled. It happened the programmer cannot do anything.

**7.** **Write about Unchecked exceptions?**

**A)** Unchecked exceptions, the programmer should handle them or through without handing . he con not simple ignore them ,as java compile will remind him something to avoid any harm caused by the rise of an exceptions .

**Long Answer Type Questions**

**8. Explain the types of errors in java?**

**A)**

**Compile Time errors:** A compile-time error happens when the program is being compiled. Generally, Compile-time errors are syntax errors, and they are caught by the compiler.

**Example:-**

Missing semicolons.

Missing brackets in classes and methods.

Misspelling of identifiers and keywords.

Missing double quotes in strings

**Run time errors:** Run-time errors occur at run-time. Generally, the program compiles but does not run correctly. Example, insufficient memory to store something or inability of the microprocessor to execute some statement comes under run-time error.

**Example:-**

\*Dividing an integer by zero.

\*Accessing an element that is out of the bound of an array.

\*Trying to store to value into an array of an in compatible

class or type.

\*Trying to cast of an instance of a class to one of it’s sub

Class.

**Logical Errors:** The programmer might be using a wrong formula or the design of the program itself is wrong. Logical errors are not detected either by Java compiler or JVM. The programmer solely responsible for them.

**9.** **How to handle the exception in java?**

**A)** When there is an exception, the user data may be corrupted. This should be tackled by the programmer by carefully designing the program by using the following three steps.

**Step 1(try ):**

The programmer should observe the statements in his program where there may be a possibility of exceptions. Such statements should be written inside a try block. A try block looks like as follows

try

{

statements.

}

**Step 2 (catch):**

The programmer should write the catch block where he should display the exception details to the user. This helps the user to understand that there is some error in the program. Catch block looks like as follows:

catch(Exceptionclass ref)

{

statements;

}

The **Throwable class** of **printStackTrace()** method fetches exception details from the exception details from the exception stack and displays them. The **‘Throwable’** class is the super class of all errors and exceptions in the java language.

**Throw** is used to force an exception by the programmer.

**Step 3 (finally):**

The programmer should perform cleanup operations like closing the files and terminating the threads in the finally block. The statements inside the finally block are executed irrespective of whether there is an exception or not. The finally block looks like as follows.

finally ()

{

Statements;

}

**10.** **Explain types of exception in java?**

**A)** Exceptions are basically two types, those are:

* 1. Checked Exceptions.
  2. Unchecked Exceptions.

**Checked Exceptions:** The Exceptions that are checked at compilation time by the java compiler are called Checked Exceptions. These Exceptions should either handle them or throw them without handling them by the programmer. Following are common checked exceptions.

Class Not Found Exception.

IOException

EOF Exception

File Not Found Exception

No Such Method Exception

1. **Unchecked Exceptions:** The exceptions which are checked by the JVM are called unchecked exceptions. These exceptions and errors and can compile the program. The programmer see their effect only when he runs the program. Following are common unchecked exceptions.

* Arithmetic Exception
* Illegal Argument Exception
* Number Format Exception
* Index Out Of Bound Exception
* Array Index Out Of Bounds Exception
* String Index Out Of Bounds Exception
* Null Pointer Exception.

**11.** **What are the built – in exceptions in Java?**

**A)**

|  |  |
| --- | --- |
| **Exception** | **Description** |
| 1. Arithmetic Exception | Arithmetic error, such as divide – by – zero . |
| 2. Array Index Out Of Bounds Exception | Array index is out – of – bounds. |
| 3. Null Pointer Exception | Invalid use of a null reference. |
| 4. Number Format Exception | Invalid conversion of a string to a numeric format. |
| 5. Security Exception | Attempt to violate security. |
| 6. String Index Out Of Bounds | Attempt to index outside the bounds of a string. |
| 7. Index Out Of Bounds Exception | Some type of index is out – of – bounds. |
| 8. Class Cast Exception | Invalid Cast. |
| 9. Class Not Found Exception | Class not found |
| 10. No Such Method Exception | A requested method does not exist. |

**12.** **Explain user define exceptions?**

**A)** Sometimes, the built-in functions in java or not able to describe a certain situation .In such cases, the user can also create his own exception, which one called “user defined exceptions”. We have to fallow the following steps to create user define exceptions.

Step1:- The user should create his own exception class as a sub class to exception class.

Step2:- The user can write his own exception class. If the user does not want to create an empty to his exception class, he can eliminate writing to default constructor.

Step3:- The user can create a parameterized constructor with a string as a parameter .He can use this to store exception details he can call super class(exception) constructor from this and send the string there.

Example:-

class MyException extends Exception

{

MyException ( )

{

}

MyException (String str)

{

super (str);

}

}

Step4:- When the user wants to raise his own exception he should create an object to his exception class and throw it using “throw” class as.

**13.** **What are the differences between error and exceptions?**

**A)**

|  |  |  |
| --- | --- | --- |
| **BASIS FOR COMPARISON** | **ERROR** | **EXCEP**  **TION** |
| Basic | An error is caused due to lack of system resources. | An exception is caused because of the code. |
| Recovery | An error is irrecoverable. | An exception is recoverable. |
| Keywords | There is no means to handle an error by the program code. | Exceptions are handled using three keywords "try", "catch", and "throw". |
| Consequences | As the error is detected the program will terminated abnormally. | As an exception is detected, it is thrown and caught by the "throw" and "catch" keywords correspondingly. |
| Types | Errors are classified as unchecked type. | Exceptions are classified as checked or unchecked type. |
| Package | In Java, errors are defined "java.lang.Error" package. | In Java, an exceptions are defined in"java.lang.Exception". |
| Example | OutOfMemory, StackOverFlow. | Checked Exceptions : NoSuchMethod, ClassNotFound. Unchecked Exceptions : NullPointer, IndexOutOfBounds. |