Unit – 1 Object Oriented Programming - CS8392

Introduction about JAVA

Java was originally developed by **James Gosling** at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems Java platform.

Object-oriented programming is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields, and code, in the form of procedures. It helps the developer by allowing for code to be easily reused by other parts of the program.

What is Java Programming?

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers **"write once, run anywhere" (WORA)**, meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java is machine independent. **JVM** is used to run byte codes on any platform.

1.Elements / Features / Concepts / Principles of OOPs

1. **Object** – Instance of class | - Run time entities which occupies memory
2. **Classes** – Collection of attributes, methods.
3. **Instance** – Obj. created at run time
4. **Inheritance** – Provides reusability |
5. **Data abstraction** – Information hiding | refers to particular feature and hiding its background details | used in s/w design phase.
6. **Encapsulation** – Binding data and method together | used in s/w implementation | Inherited
7. **Polymorphism** –Ability to take more than one form | Types: compile time & run time
8. **Message passing** – An object sends data to another obj.

2.Characteristics of Java

1. **Simple** – No pointer concept so easy to debug – auto memory allocation & De-allocation
2. **Portable** – JVM | run in any platform
3. **Object Oriented** – Almost everything in java is object – it is an entity has attributes, functions to manipulate.
4. **Platform independent** – WORA | write once, Run anywhere - JVM
5. **Dynamic and distributed** – java classes can be distributed in networks – java.net package.
6. **Multithreaded** – concurrency (run multiple pgm at same time) – Parallel execution – built in thread class.
7. **Robust** and **secure** – good exception handling, explicit methods – array bound checking
8. **Interpreted language** –source code stored in .java – compiled file in .class(bytecode) – JVM interprets and executes the program.
9. **High performance** – Byte codes are highly optimized | JVM executes it faster
10. **Architecture-neural** – Independent of hardware

3.JDK tools – Java Development Kit

1. **Applet viewer** – Enables to run java applets(without browser).
2. **Java** – Java interpreter – runs applets and appln. by reading and interpreting byte code
3. **Javac** – Java compiler – converts source code to byte code
4. **Javadoc** – Creates HTML format documentation
5. **Javah** – Java header – Produces header files to use methods
6. **Javap** – Java disassemler - Convert bytecode to program description
7. **Jdb** – Java debugger – to find errors

4.JRE – Java Runtime Environment

The JRE provides the minimum requirements for executing a Java application
JRE = JVM + Java packages classes + run time lib.

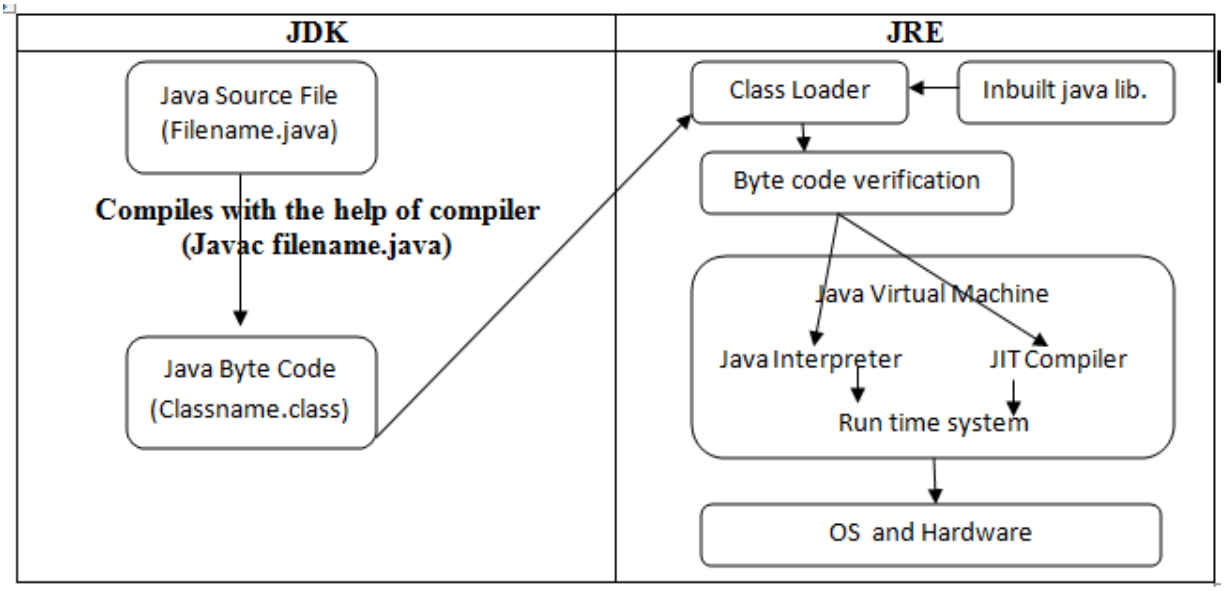
5.JVM – Java Virtual Machine

- Java source code is saved with the extension .java.
- When compiled .class file created. | .class file consists of byte codes. This makes java machine independent.
- Java byte codes are understandable by JVM.
- JVM is responsible for garbage collection, etc., JVM is platform dependent.

6.Java source file structure

Documentation section	-
Package stmt	-
Import stmt	-
Interface stmt	-
Class defn.	-

Main() method class
{
Definitions;
}
	-



Comments:

- Comments are used to give overview of codes. Ex: /* Main function */ or //

Variable:

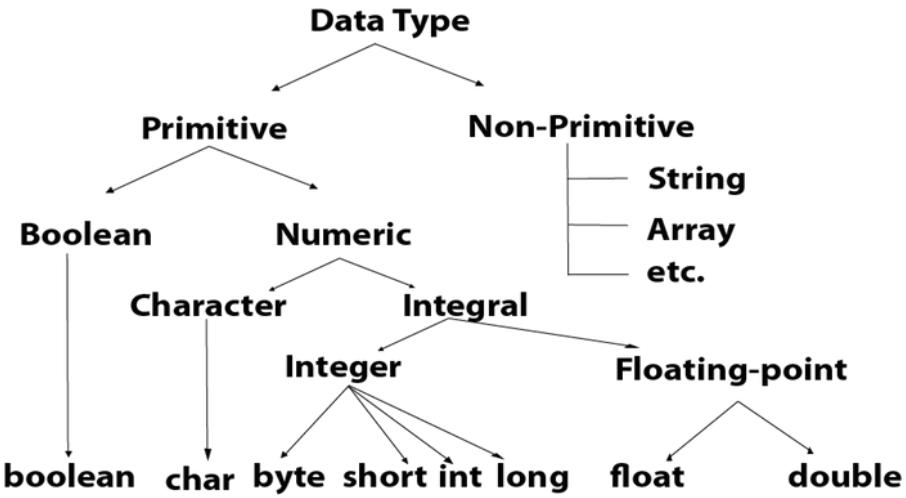
- Name of memory location in RAM. Ex: int 10;

Data type:

Two types: i. Primitive data types

(Eight types – Byte "8-Bit, Short "16-B, int "32-B, long "64-B, float "32-B, double "64-B, char "16-B, Boolean "True or False)

ii**. Reference data types** –Used in arrays.



Tokens – Smallest individual and logical unit of java statement

i. Reserved keywords ii. Identifiers iii. Literals iv. Operators v. Separators

- Identifiers – Kind of token defined by programmer. Ex: int a;
- Literals – Stores sequence of characters
- Separators – ( ) , { }
- Constants – fixed values do not change during execution

Keywords

Pre defined. 50 reserved words. Ex: class, return, do..

Operators:

1. Assignment Operators| =
2. Arithmetic Operators| + , -
3. Relational Operators | = = , <=
4. Boolean logical Operators | &&, ||, !
5. Conditional Operators | ?:
6. Type conversion Operators | Obj. reference

Two types 1.Implicit casting(Widening) b-s-i-l-f-d 2.Explicit casting(Narrowing) d-f-l-i-s-b

7. Bitwise and bit shift Operators | >>>, ^, ~
8. Increment and decrement Operators | ++, - -

7.Control flow / statements / structures

i. Decision making / Selection stmt
   i. If statement
   ii. If else
   iii. Switch case

ii. Loops
   i. For loop
   ii. While loop
   iii. Do.. while loop

8.Arrays

- Collection of similar type of elements. – Group of elements stored under common name.

- Array in java is index-based, the first element of the array is stored at the 0 index.

Ex: int a[ ]; To allocate memory, a = new int[10];

Types: i. One dimensional array | Ex: int age[3];

ii.Two dimensional array | Ex: int num[3] [4];
iii.Multi dimensional array | Ex: int num[ ][ ][ ][ ];

Example:

```
class onedimarray
{
        public static void main(String args[])
        {
        //dec. and instantiation
        int arr[ ] = {10,20,30,40,50};
        //printing array
        for(int i=0; i<arr.length; i++)
        System.out.println(arr[i]);
        }
}
```

```
class twodimarray
{
        public static void main(String args[])
        {
        int arr[ ][ ] = {{1,2,3},{4,5,6},{7,8,9}};
                for(int i=0; i<3; i++)
                {
                        for(int j=0; j<3; j++)
                        {
                        System.out.print(arr[i][j]+"  ");
                        }
                System.out.println();
                }
        }
}
```

Advantages

- **Code Optimization:** It makes the code optimized, we can retrieve or sort the data efficiently.

- **Random access:** We can get any data located at an index position.

Disadvantages

- **Size Limit:** We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically.

9. Constructors

- **Constructor** is a block of code that **initializes** the newly created **object**. It is special method called automatically when an object is created. Class name and constructor name is same.

Types: i. Default constructor
ii.Parameterized constructors

Properties of constructors

i. No return type. No void.
ii. Must be declared in public
iii. Cannot be virtual
iv. Cannot be inherited
v. Constructors in Java do not return anything explicitly. implicitly, they **return** the current instance of the class whose **constructor** it is.

Ex: Default constructor

```
class box
{
        double w, h, d;
        box()
        {
                System.out.println("Constructing box");
                w=10;
                h=10;
                d=10;
        }
        double volume()
        {
                double v;
                v=w*h*d;
                System.out.println("volume="+v);
                return 0;
        }
}
class def_const1
{
        public static void main(String args[ ])
        {
                box b1 = new box();
                b1.volume();
        }}
```

Ex: Parameterized constructor

```
class box1
{
        double w, h, d;
        box1(double wi, double ht, double de)
        {
                w=wi;
                h=ht;
                d=de;
        }
        double volume()
        {
                double v;
                v=w*h*d;

        System.out.println("volume="+v);
                return 0;
        }
}
class const_param1
{
        public static void main(String args[])
        {
            box1 b2 = new box1 (10,20,30);
            b2.volume();
        }
}
```

Return keyword is used to exit from method.

Destructor: No destructor in java. **Garbage collector** in java performs i) freeing up memory allocated for objects and ii) Cleaning up resources.

Call by value: calling a method with parameter as value. The changes done to the parameter don't reflect in caller's scope.

Call by reference: calling a method with a parameter as a reference. The changes done to the parameter reflect in caller's scope. **Java uses only call by value.**

10.Creating class and objects

- **Class** is a template/blueprint that describes behaviors of object. | Classes stored in heap. | Classes are Reference type.
- **Object** is an instance of class. "**New**" keyword used

Ex:

```
Class Demo
{
Psvm..
{
        Demo d1; // Creating reference obj
        d1= new Demo(); //Object for Demo
}
}
```

11.Methods

- In java method is equivalent to function.
- Every method must be declared with in class.
- Ex: isEmpty | compareTo | run

Using parameter / Specifying method arguments

A method can have zero or more arguments is called parameters.

Two ways:

1. **Specifying multiple arguments**

```
public void empdetails(String name, int age)
{
Ename = name;
Eage = age;
}
```

2. **Specifying no arguments** – if no arguments leave it empty

What is JAR?

A **JAR** (Java Archive) is a package file format typically used to aggregate many Java class files and associated metadata and resources (text, images, etc.) into one file to distribute application software or libraries on the Java platform.

12. Access specifier's:

Java Access Specifiers (also known as Visibility **Specifiers**) regulate **access** to classes, fields and methods in **Java**. These **Specifiers** determine whether a field or method in a class, can be reused or not. **Private is most restrictive**

1. Public
2. Protected
3. Default
4. Private

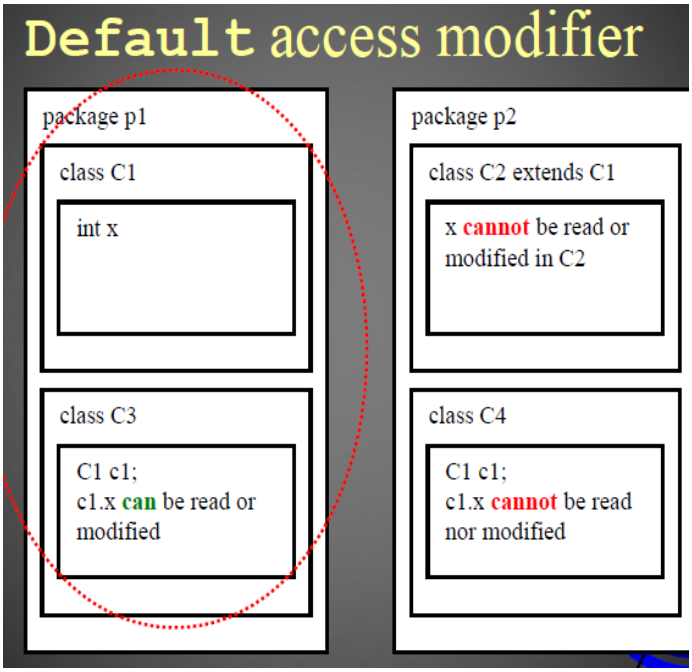
Rules:

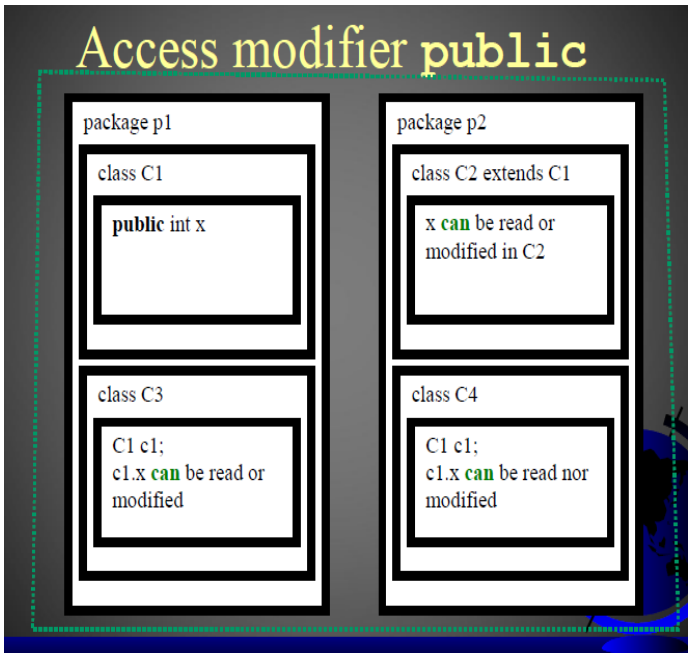
Access Location	Access Modifier			
	Public	Protected	Default	Private
Same class	Yes	Yes	Yes	Yes
Sub class in same package	Yes	Yes	Yes	No
Other classes in same package	Yes	Yes	Yes	No
Subclass in other packages	Yes	Yes	No	No
Non-subclass in other package	Yes	No	No	No

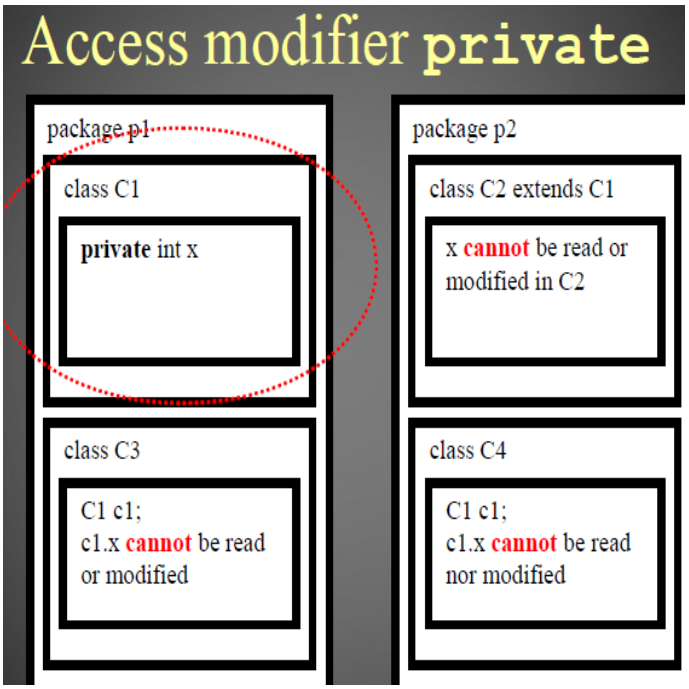
Example:

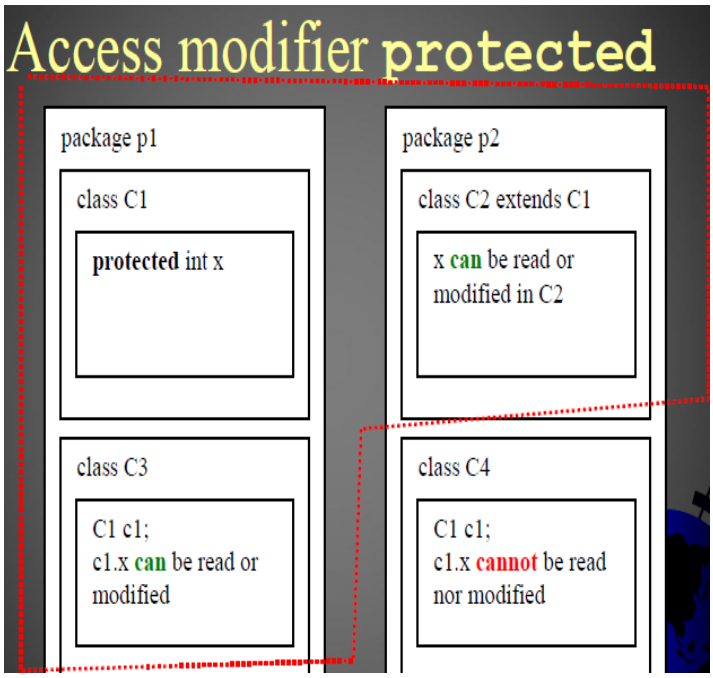
```
package p1;
public class class1
{
public int a;
int b;
private int c;

public void fun1()
        { s.o.p(a+b+c);
        }

void fun1()
        { s.o.p(a+b+c);
        }

private void fun1()
        { s.o.p(a+b+c);
        }
}
public class class2
{       p.s.vm..{
        class1 obj = new class1();
        obj.a; //allowed
        obj.b; //allowed
        obj.c; //cant access

        obj.fun1(); //allowed
        obj.fun2(); //allowed
        obj.fun3(); //cant access
         }}
```

```
import p1.*;
public class class3
{
psvm…{
class1 obj = new class1();
obj.a; //allowed
obj.b; //cant access
obj.c; //cant access
obj.fun1(); //allowed
obj.fun2(); //cant access
obj.fun3(); //cant access
}
}
```


Default access modifier
package p1
class C1
int x
class C3
C1 c1;
c1.x can be read or modified
package p2
class C2 extends C1
x cannot be read or modified in C2
class C4
C1 c1;
c1.x cannot be read nor modified


Access modifier public
package p1
class C1
public int x
class C3
C1 c1;
c1.x can be read or modified
package p2
class C2 extends C1
x can be read or modified in C2
class C4
C1 c1;
c1.x can be read nor modified


Access modifier private
package p1
class C1
private int x
class C3
C1 c1;
c1.x cannot be read or modified
package p2
class C2 extends C1
x cannot be read or modified in C2
class C4
C1 c1;
c1.x cannot be read nor modified


Access modifier protected
package p1
class C1
protected int x
class C3
C1 c1;
c1.x can be read or modified
package p2
class C2 extends C1
x can be read or modified in C2
class C4
C1 c1;
c1.x cannot be read nor modified

13.Static members

- The memory of static fields will be stored in constant pole instead heap organization.
- Only one copy of memory created and shared
- The default value of static field is zero.

3 ways 1) Static field – Using "this" keyword
2) Static method – Using static keyword
3) Static class – Using static keyword in nested class

Ex: Static Field Program

```
class stat
{
        int x;   static int y;
        void set_value(int x, int y)
        {
                this.x=x; //instance variable and static field
                this.y=y;
        }
        void print()
        {
        system.out.println("the value of x:" +x);
        system.out.println("the value of y:" +y);
        }}
class stat_main
{
        public static void main(string args[])
        {
        stat s1 = new stat();
        s1.set_value(10,20);
        s1.print();

        stat s2 = new stat();
        s2.print(); }}
```

Ex: Static method

```
class staticdemo
{
        static int age=37;
        static int height=147;
        static class inner
        static void display()
        {
                System.out.println("Age ="+age);
                System.out.println("Height ="+height);
        }
public static void main(String args[])
{
        staticdemo.inner.display(); } }
```

Ex: static nested class

```
class outer
{
        static int a=10;
        static class inner
        {
                void msg()
                {
                System.out.println("Value=" +a);
                }
        }
public static void main(String args[])
        {
        outer.inner obj1= new outer.inner();
        obj1.msg(); //msg() is not static so creating object
        }
}
```

14.Packages

Definition: Packages is a mechanism in which variety of classes and interfaces can be grouped together.

Advantages:

i. Code reused – from other package
ii. Same name – two classes from two different packages
iii. Possible to hide the classes
iv. Name of the directory becomes the package name.

Two types:

i. Built-in packages

Ex: java.lang, java.util, java.io, java.awt, java.applet

ii. User defined packages

Ex: Syntax to create package:

```
package package_name;

package abc;
```

Syntax to Import package:

```
import abc.*;
```

Example:

STEP 1: Create a folder p1 and follow the below code,

```
package p1;
public class testpackage
{
        public void display()
        {
        System.out.println("Hi");
        }
}
```

Save the file in p1 folder as "testpackage" and compile it.

STEP 2: Come out from the folder p1 and save the below code as "test".

```
import p1.*;
public class test
{
        public static void main(String args[])
        {
                testpackage tp = new testpackage();
                tp.display();
        }
}
```

Now compile and run the "test" file

15. JavaDoc comments

- A convenient and standard way to document java code. Creates HTML format documentation.

Two types:

i. Class level comments
ii. Member level comments

Class level comments:

- It Provides description and purpose of the class.

```
/**
*@author ABC
*The Student class contains marks
*/
class Student
        {
        //Student class code
        }
```

Member level comments:

- It describes about data members, methods, and constructors.

Tags	Description
@author	Author name
@since	To show from when used
@version	Current version
@deprecated	Deprecated should not be used
@param	Describes name of the method
@return	Return type
@throws	Type of action
@exception	Error

To generate Javadoc comments

Go to the directory where you have java files. Then

D:\17CS01> **Javadoc – author <filename>.java**

Assignment 01:

Part A: | 2 Marks

1. Mention some of the separators used in java programming.
2. How dynamic initialization of variables is achieved in java? Ex.
3. What is JVM, JDK, JAR?
4. List 5 features of java.
5. Define objects and classes in java.
6. D/B structure and class.
7. D/B static and non static variables.
8. Define encapsulation.
9. What is abstract class?
10. What do you mean by instance variable?
11. D/B constructor and method.
12. What is meant by parameter passing constructors? Ex.
13. What is package?
14. Enumerate two situations in which static methods are used.
15. Mention the necessity for import statement.
16. What is a token? List types.
17. D/B break and continue statement.
18. List benefits of encapsulation.
19. What is API package?
20. Example program for "while" and "do-while".

Part B | 13 Marks

1. Briefly explain elements of object oriented programming.
2. What are JDK, JRE and JVM? Explain JDK tools.
3. Explain structure of java program.
4. Explain constructor types with example program.
5. Explain one and two dimensional array with example program.
6. Explain access specifiers with example.
7. List packages types. Write an example program to define and import package.
8. What is javadoc? List 5 tags and description with example program.
9. What are all static members? Explain with example.
10. Write down the step to compile and run a java program.

Part C – Programs

1. Write a java program for computing Fibonacci series.
2. Write a java program that reverses the digits of given number.
3. Write a java program to find factorial
4. Write a java program to display any number pattern.
5. Write a simple java program using
    a. if statement
    b. if-else
    c. while
    d. do-while
    e. switch
    f. for loop
    g. Boolean
    h. Type conversion operator