Day 8 Assignment - 23-04-23

CONSTRUCTOR:-

- A constructor in java is a special method that is used to initialize objects
- The constructor is called when an object of the class is created.
- It can be used to set initial values for object attributes.
- Every time an object is created using the new() keyword, atleast one constructor is called.

RULES:-

- Constructors must have the same name as the class name.
- constructors do not have any return type.
- constructors are called only once at the time of object creation.

Uses of constructors:-

1. It is used to initialize the values at the start of execution.

2. It calculates needed memory for variables and methods in class and provide information to JVM to calculation of Memory.

SYNTAX:

```
class classname
{ classname() // constructor
{ statement;
}}
```

Types of constructors in java:

- No argument constructor
- Parameterized constructor

No argument constructor:

A constructor that has no parameter is known as default constructor If we don't define a constructor in a class, then the compiler creates a default constructor (with no arguments for the class. And if we write a constructor with arguments or no arguments then the compiler does not create a default Constructor

Parameterized Constructor:

A constructor that has parameter is known as parameterized constructor.

If we want to initialize fields of the class with our own values, then use a parameterized constructor.

CONSTRUCTOR OVERLOADING:

Constructor Overloading in java is a technique of having more than one constructor with different parameter list.

They are arranged in a way tha

CONSTRUCTOR CHAINING:

Constructor chaining is the process of calling one constructor from another constructor with respect to current object.

Constructor chaining can be done in two ways.

1.) within same class: It can be done using this() keyword for constructors in the same class.

2.) from base class: by using super() keyword to call constructor from the base class.

Example program:

```
package org.abstraction;
public class ConstructorExample
{ ConstructorExample()
  { System.out.println("No
      caugument constructor");
  }
ConstructorExample(int a)
{ System.out.println("1 arg
      constructor");
}
ConstructorExample(String name,
                   char i)
{ System.out.println("2
      Arg constructor");
}
```

```
public static void main (String [] args)
{ ConstructorExample c = new ConstructorExample();
  ConstructorExample c1 = new ConstructorExample(4);
  ConstructorExample c2 = new ConstructorExample("Java", '1');
}
}
```

~~Shaik~~

Encapsulation :: (POJO class - plain old java object)

Encapsulation in java is a mechanism of wrapping the data (variables) and code (methods) together as a single unit.

In encapsulation, the variables of a class will be hidden from other class and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

To achieve encapsulation in java.

Declare the variables of a class as private.

Provide public setter and getter to modify and view the variable values.

```
package org.encapsulation.
public class Mark{
    Private static int sem_mark=64;
    public int getsem_mark(){
        return sem_mark;}
public void setsem_mark(int mark)
  {     this.sem-mark = mark;
        System.out.println("Updated Mark:"+
                           this.sem_mark);
  }
}
package org.encapsulation.
public class College extends Mark{
  public static void main(String[]args)
    {College student = new College();
     College staff = new College();
     System.out.println(student.getsem_mark);
     staff.setsem_mark(77);
}}
```

Benefits of Encapsulation:

The fields of a class can be made read-only or write only.

A class can have total control over what is stored in its fields.

operators:

1) Unary operators

Postfix:

Exp++

int a = 10;

a++ → a+1

c = a++ (15) → c=10 → a=a+1 → 11

c=11 → a=a+1 → 12

c=12 → a=12+1 → 13

c=13 → a=13+1 → 14

c=14 → a=14+1 = 15

Prefix:

++Exp C=11 a=a+1
C=++a C=12 a=12+1
C=13 a=12+1
C=14 a=14+1
C=15 ...

.....

2. Arithmetic operator:

+ - * / %

a=10/2=5
a=10%2=0
a=10%3=1

3) Relational operator.

< > ≤ ≥ = ≠

a=100, b=200

a<b 100<200 true
a>b 100>200 false
a≤b 100≤200 true
a≠b 100≠200 false
a==b 100==200 true

Difference between = and ==

= → Assignment operator, a=10; b=20; C=30;
== → Comparison operator, age=18, age==18;
doc=voterid; adhar==voterid voterid
==voterid

4) logical operator:

** Logical AND → Exp1 and Exp2 must be true.

exp1 && exp2, 100=100 && 2≥2
true && true true, 100≠100 && 20≥2
false && true false.

t && t True
t && f false
f && T False
f && f false

|| - logical OR → Exp1 or Exp2 either one shall be true.

t || t True
t || f True
f || t True
f || f false.

! logical NOT
exp true result true
exp false result false

5.) Assignment operator :-

+=, -=, *=, /=, %=

+= ⟶ a = 10; b = 20; b+=a; 20+=10 b=3

-= → b-=a; 20-10; b = 10

= → b=a; 20*10 b = 200

/= → b/=a; 20/10 b = 2

%= ⟶ b%=a; 20%10 b = 0

6.) Ternary operator : ? :

Syntax:

a>b ? a : b

NOTES TAKING

23-4-23

constructor:.

→ Initialize the value
→ Initialize default values

Obj creation:-

classname objref = new Constructor();

constructor chaining:.

Task:. this and super.

Encapsulation :- / Pojo class

Data hiding

```
Private int sem_mark = 64;
public int getSem_mark()
{
    return Sem_mark;
}
```

```
System.out.println(student.getSem_mark());
```

```
public void setsem_mark()
{
}
```

Sure → getters and setters.