16.04.2023
Assignment - Day 6

POLYMORPHISM :: (Executing methods in more than one form)

The word polymorphism means having many forms.

Polymorphism allows us to perform a single action in different ways

Poly - many, Morphism - forms

when there are multiple functions with the same name but different types or number of parameters then the functions are said to be overloaded

Two types of polymorphism :-

Method overloading (compile-time polymorphism) - static Binding)
Method Overriding (runtime polymorphism)
Dynamic binding.

-Method Overloading :-

Two or more methods may have the * same name if they differ in parameters, different types of parameters, or both). These methods are are called overloaded methods and this feature is called method overloading.

Rules of Method Overloading :-

- Same class
- Same method name
- Diff argument
  - data type
  - Order
  - number

Example Program

```
Package com.polymorphism;
Public class StudentDetails
{
  public void formFill()
  { System.out.println("No Arguments
    method");
  }
  Public void formFill(String name)
  { System.out.println("1 argument
                        method")}
  public void formFill(String name, int num
  { System.out.println("2 arguments
                        method");

  Public void formFill(String name, int
                       num1, int num2)
  { System.out.println("3 arg methods");
  }
  Public static void main(String []
                          args)
  {
    StudentDetails sd = new student
                        Details();
    sd.formFill("java", 9);
    sd.formFill(3, "Satyam");
    sd.formFill();
    sd.formFill("abi", 123, 456);
    sd.formFill("Java");
```

}}

Example Program using "this" keyword
(this points to current class)

```
Package com.polymorphism;
  Public class StudentDetails {
      public void formFill()
      { this.formFill("Saleniun");
        System.out.println("No arguments
                method");
      }
      public void formFill(String name)
      { System.out.println("1
          arg method");}
      Public void formFill(String name,
              int num1, int num2)
      { this.formFill();
        System.out.println("2arg
                method");
      }
    Public Static void main
          (String [] args)
    {StudentDetails sd = new
          StudentDetails();
      sd.formFill("java", 8);
}
```

METHOD OVERRIDING:

- Declaring a method in sub class which is already present parent class is known as method overriding.

Overriding is done so that child class can give its own implementation to a method which is already provided by parent class.

In this case the method in parent class is called overiden method and in child is called is called Overriding method.

if subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.

Rules of Method Overriding:

- Same method name.
- Same Arguments.
- Diff class.

- Example program:

```
package com.polymorphism;

public class RBI {
    public void accountopening()
    { System.out.println("Account
      Open Fee - 2000"); }
```

```
public void fixedDeposit()
 { System.out.println("Interest Rate-8%");
}
public void recurringDeposit()
  { System.out.println("Interest Rate
                        11%");
}
public void insurance()
  { System.out.println("Insurance");
  }
}
package com.polymorphism;
  public class HSBC extends RBI
  { public void accountOpening()
   { System.out.println("HSBC-Account
                  open fee-5000");
   }
   public void fixedDeposit()
     { System.out.println("HSBC
                Interest Rate-13%");
     }
   public void loans()
   { System.out.println("HSBC Interest
  Rate-15%"); }
public static void main(String
                  [] args)
  { HSBC h = new HSBC();
        h.accountOpening();
        h.fixedDeposit();
        h.insurance();
        h.loans();
```

h.recurringDeposit();
r.insurance();
}}

Advantage of method overriding:-

The main advantage of method overriding is that the class can give its own specific implementation to a inherited method without even modifying the parent class code.

Using super keyword:-

```
package com.polymorphism;
public class HSBC extends RBI
{ public void amountOpening(){
    super.amountOpening();
    System.out.println("HSBC-
  Account open fee-5000");
}
public void fixedDeposit()
{ System.out.println("HSBC-Interest
                    Rate-10%.");
}
public void recurringDeposit()
{System.out.println("HSBC Interest
                    Rate 15%");}
public void loans()
{System.out.Println("HSBC-Interest rate
                    15%.");
}
public static void main(String[] args)
    { HSBC h=new HSBC();
      h.amountOpening();
```

h.fixedDeposit();
h.insurance();
h.loans();
h.recurringDeposit();
r.insurance();
}}

STATIC BLOCK and STATIC METHOD:-

STATIC BLOCK:-

This code inside the static block is executed only once: the first time the class is loaded into memory.

Static block will executed first before main method execution.

Syntax:

```
static
{ // statement 1;
}
```

MULTIPLE static BLOCK

Eg:,

```
public class StaticBlockExample{
    static {
        System.out.println
            ("Hai everyone Welcome");
    }
{ System.out.println ("Instance Block");
}
public static void main(String
        [] args)
```

{ StaticBlockExample sb = new
StaticBlockExample();
}}

Static method:

Static methods belong to the class and they will be loaded into the memory along with the class, you can invoke them without creating an object (using the classname as reference).

SYNTAX:

public static void methodname();
{ }

Restriction on static method:

* you cannot access a non-static member (method or, variable) from static context.
* This and super cannot be used in static context.
* The static method can access only static type data (static type instance variable.
* you cannot override a static method you can just hide it.

Static modifier:

It is applicable for methods and variables but not for classes.

static methods can only access the static members of the class and can only be called by other static methods.

Example program:

```
package com.polymorphism;
public class NonAccessModifierExample {
    int a = 10;
    int b = 20;
    static int c = 20;
    int d;
    static int e;
  public void norMethod()
   { d = a+b;
     System.out.println("normal method"
                        + d);
  }
  public static void StaticMethod()
  {  int a = 10+20; //implicitly it will
                      take it as static
     // e = a+b; //we can't use non
                   static member in static field.
     System.out.println(e);
  }
  public static void main(String [] args)
   { NonAccessModifierExample n =
       new NonAccessModifierExample();
     n.norMethod();
     StaticMethod(); // with in the class
     NonAccessModifierExample.StaticMethod();
                        //outside the class.
  }
```

Access modifiers (static and final)

final:

It is applicable to classes, methods and variables.

If we declare a class as final we can't inherit that class (i.e., we can't create child class)

If we declare a method as final we can't we can't Override a method

if we declare a variable as final we must do the initialization we cannot change the value.

Syntax:

final class

```
final class classname{
}
```

final method

```
public final void methodname()
{ }
```

final variable

```
final datatype variablename
        = value;
```

Notes taking: - 16-4-2023 - Day 6 -

① 2 arg method

~~② 1 arg method~~

② 2 arg method but dif order

③ No argument

④ 3 arg method

⑤ 1 arg method

chaining Method:

this and super keywords.

this points to Parent class

using with this class.

method overloading:

Compile time polymorphism

(or)

Static binding

dynamic way of object creation

Static block:

Non Access Modifier example

static { } Block

execute Before main (or) without main.

```
static {
    }
main ()
    { }
```

{ } Instance Block.

By creating Instance

Static method :-

only static variable or used is static method.

local variable are used in static variable

final class is final