JBK1007-Assignment –Constructor & Superthis

Program for constructor Demo

```
public class Constructor_Demo {
        public Constructor_Demo()     //Default Constructor {
                System.out.println("Constructor Running");
}
        public Constructor_Demo(int a)      //Parameterized
Constructor {
                System.out.println("Constructor Running" + a);
 }
        public static void main(String[] args) {
Constructor_Demo cd=new Constructor_Demo();//default
constructor calling
Constructor_Demo cd1=new Constructor_Demo(5);
//parameterized constructor calling
}       }
```

Program for constructor chaining Demo

```
public class ChainingDemo {
   //default constructor of the class
       public ChainingDemo(){
       System.out.println("Default constructor");
   }
     public ChainingDemo(String str) {
       this();
        System.out.println("Parametrized constructor with single
param");
   }
     public ChainingDemo(String str, int num) {
       //It will call the constructor with String argument
        this("Hello");
        System.out.println("Parametrized constructor with double
args");
   }
    public ChainingDemo(int num1, int num2, int num3 {
        // It will call the constructor with (String, integer)
arguments
      this("Hello", 2);
 System.out.println("Parametrized constructor with three
args");
   }
```

```
    public static void main(String args[]) {
      //Creating an object using Constructor with 3 int arguments
      ChainingDemo obj = new ChainingDemo(5,5,15);
} }
```

Program for constructor Overloading with Getter ,Setter with this keyword

```
public class StudentData{
  private int stuID;
  private String stuName;
  private int stuAge;
    StudentData()  {
     //Default constructor
     stuID = 100;  stuName = "New Student";
     stuAge = 18;
  }
     StudentData(int num1, String str, int num2) {
     //Parameterized constructor
     stuID = num1;   stuName = str;   stuAge = num2;
  }
//Getter and setter methods
     public int getStuID() {
     return stuID;
  }
    public void setStuID(int stuID) {
     this.stuID = stuID;
  }
   public String getStuName() {
     return stuName;
  }
   public void setStuName(String stuName) {
     this.stuName = stuName;
  }
   public int getStuAge() {
     return stuAge;
  }
  public void setStuAge(int stuAge) {
     this.stuAge = stuAge;
  } }
class TestOverloading{
  public static void main(String args[])  {
     //This object creation would call the default constructor
```

```
      StudentData myobj = new StudentData();
      System.out.println("Student Name is:
"+myobj.getStuName());
      System.out.println("Student Age is: "+myobj.getStuAge());
      System.out.println("Student ID is: "+myobj.getStuID());
      /*This object creation would call the parameterized
       * constructor StudentData(int, String, int)*/
   StudentData myobj2 = new StudentData(555, "Chaitanya", 25);
      System.out.println("Student Name is:
"+myobj2.getStuName());
      System.out.println("Student Age is: "+myobj2.getStuAge());
      System.out.println("Student ID is: "+myobj2.getStuID());
  }  }
```

Program for constructor with this()

```
public class ConstOverloading  {
   private int rollNum;
   ConstOverloading()  {
     rollNum =100;
   }
ConstOverloading(int rnum)  {
     this();
     /*this() is used for calling the default
      * constructor from parameterized constructor.
      * It should always be the first statement
      * in constructor body.
      */
     rollNum = rollNum+ rnum;
   }
     public int getRollNum() {
           return rollNum;
   }
    public void setRollNum(int rollNum) {
           this.rollNum = rollNum;
   }
}
class TestDemo {
   public static void main(String args[]){
      ConstOverloading obj = new ConstOverloading(12);
      System.out.println(obj.getRollNum());
    } }
```

Program for super variable

```
//Parent class or Superclass
class Parentclass{
  int num=100;
} //Child class or subclass
class Subclass extends Parentclass{
  int num=110;
  void printNumber() {
    //Super.variable_name
    System.out.println(super.num);
  }
public static void main(String args[]){
    Subclass obj= new Subclass();
    obj.printNumber();
  }  }
```

Program for Child class Constructor call parent class constructor implicitely

```
class Parentclass{
  Parentclass() {
    System.out.println("Constructor of Superclass");
  } }
class Subclass extends Parentclass{
  Subclass(){
        /* Compile adds super() here at the first line
         * of this constructor implicitly
         */
        System.out.println("Constructor of Subclass");
  }
Subclass(int num){
        /* Compile adds super() here at the first line
         * of this constructor implicitly
         */
        System.out.println("Constructor with arg");
  }
void display(){
        System.out.println("Hello");
  }
public static void main(String args[]){
        // Creating object using default constructor
        Subclass obj= new Subclass();
        //Calling sub class method
```

```
      obj.display();
      //Creating object 2 using arg constructor
      Subclass obj2= new Subclass(10);
      obj2.display();
  }  }
```

call super() explicitly too

```
class Parentclass{
  Parentclass(){
        System.out.println("Constructor of Superclass");
  }  }
class Subclass extends Parentclass{
  Subclass(){
        /* super() must be added to the first
         * line of constructor otherwise it would
         * throw compilation error:
         * " Constructor call must be the first statement
         * in a constructor".
         */
        super();
        System.out.println("Constructor of Subclass");
  }
void display(){
        System.out.println("Hello");
  }
 public static void main(String args[]){
        Subclass obj= new Subclass();
      obj.display();
  } }
```

Program for calling super class method using super

```
class Parentclass{
  void display() {
        System.out.println("Parent class method");
  }  }

class Subclass extends Parentclass{
  void display(){
        System.out.println("Child class method");
  }
void printMsg(){
        //This would call Overriding method
        display();
        //This would call Overridden method
        super.display();
  }
 public static void main(String args[]){
        Subclass obj= new Subclass();
        obj.printMsg();
  } }
```

Singleton Class Example Using Private Constructor

```
public class MySingleTon {
     private static MySingleTon myObj;
   /**
    * Create private constructor
    */
   private MySingleTon(){
        }
   /**
    * Create a static method to get instance.
    */
   public static MySingleTon getInstance(){
      if(myObj == null){
         myObj = new MySingleTon();
      }  return myObj;
   }
 public void getSomeThing(){
      // do something here
      System.out.println("I am here....");
   }
public static void main(String a[]){
      MySingleTon st = MySingleTon.getInstance();
      st.getSomeThing();
   }}
```

Program for super variable

```
//Parent class or Superclass
class Parentclass{
  int num=100;
}
//Child class or subclass
class Subclass extends Parentclass{
  int num=110;
  void printNumber(){
    //Super.variable_name
    System.out.println(super.num);
  }
public static void main(String args[]){
    Subclass obj= new Subclass();
    obj.printNumber();
  }  }
```

Program for Child class Constructor call parent class constructor implicitely

```
class Parentclass{
  Parentclass(){
    System.out.println("Constructor of Superclass");
  } }
class Subclass extends Parentclass{
  Subclass(){
       /* Compile adds super() here at the first line
        * of this constructor implicitly
        */
       System.out.println("Constructor of Subclass");
  }
Subclass(int num){
       /* Compile adds super() here at the first line
        * of this constructor implicitly
        */
       System.out.println("Constructor with arg");
  }
void display(){
       System.out.println("Hello");
  }
public static void main(String args[]) {
       // Creating object using default constructor
       Subclass obj= new Subclass();
       //Calling sub class method
     obj.display();
     //Creating object 2 using arg constructor
     Subclass obj2= new Subclass(10);
     obj2.display();
 }  }
```

call super() explicitly too

```
class Parentclass{
  Parentclass() {
       System.out.println("Constructor of Superclass");
  } }
class Subclass extends Parentclass{
  Subclass() {
       /* super() must be added to the first
        * line of constructor otherwise it would
        * throw compilation error:
        * " Constructor call must be the first statement
        * in a constructor".
        */
       super();
       System.out.println("Constructor of Subclass");
  }
void display() {
       System.out.println("Hello");
  }
public static void main(String args[]) {
       Subclass obj= new Subclass();
      obj.display();
  } }
```

Program for calling super class method using super

```
class Parentclass{
  void display(){
       System.out.println("Parent class method");
  }  }
class Subclass extends Parentclass{
  void display(){
       System.out.println("Child class method");
  }
void printMsg(){
       //This would call Overriding method
       display();
       //This would call Overridden method
       super.display();
  }
public static void main(String args[]){
       Subclass obj= new Subclass();
       obj.printMsg();
  } }
```