Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

SAIRAM DIGITAL RESOURCES

YEAR II
SEM III

CS8392

OBJECT ORIENTED PROGRAMMING
(Common to CSE, EEE, EIE, ICE, IT)

UNIT NO 1

INTRODUCTION TO OOP AND JAVA FUNDAMENTALS

1.5 Constructors, Methods

COMPUTER SCIENCE & ENGINEERING

Class, Object and Methods

```
//RectArea.java
class Rectangle
{
int length, width;
void getData(int x,int y)
  {
    length=x;
    width=y;
  }
int rectArea()
  {
   int area=length*width;
  return(area);
   }
}
```

```
class RectArea
{
public static void main(String args[])
 {
int area1;
Rectangle rect1=new Rectangle();
rect1.length=15;
rect1.width=10;
area1=rect1.length*rect1.width;
System.out.println("Area1 " + area1);
}
}
```

Accessing Instance Variables and Methods:

Instance variables and methods are accessed via created objects.

To access an instance variable the fully qualified path should be as follows:

/* First create an object */

Syntax:

ObjectReference = new Constructor();

Example:

Rectangle rect1=new Rectangle();

/* Now call a variable as follows */

Syntax:

ObjectReference.variableName;

Example:

rect1.length=15;

/* Now you can call a class method as follows */

Syntax:

ObjectReference.MethodName();

Example:

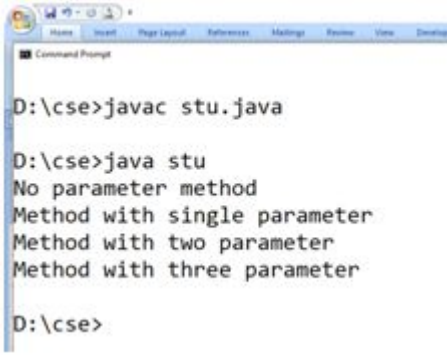
rect2.getData(20,12);

Method Overloading

- Method overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different.
- It is similar to constructor overloading in java

Sample Program

```
class stu
{
void display()
{
  System.out.println("No parameter method");
}
void display(int a)
{
     System.out.println("Method with single parameter");
}
void display(int a,int b)
{
     System.out.println("Method with two parameter");
}
void display(int a, int  b, int c)
{
     System.out.println("Method with three parameter");
}
public static void main(String a[])
{
    stu s1=new stu();        // No parameter
    s1.display();
    stu s2=new stu();        // single
    s2.display(1);
    stu s3=new stu();        // two
    s3.display(1,2);
    stu s4=new stu();        // three
    s4.display(1,2,3);
}
}
```



Method Overloading

```
public class student
{
int rno;
String name;
int age;
void display(int r)
{       rno=r;
         System.out.println("Rno     "+rno);
}
void display(int r, String n)
{       rno=r;
        name=n;
        System.out.println("Rno     "+rno);
         System.out.println("Name    "+name);
}
void  display(int r, String n, int a)
{       rno=r;
        name=n;
        age=a;
        System.out.println("Rno     "+rno);
         System.out.println("Name   "+name);
        System.out.println("Age     "+age);
}

public static void main(String a[])
{
        student s1=new student();
        student s2=new student();
        student s3=new student();
        System.out.println("first method");
        s1.display(11,"sai");
        System.out.println("second method");
        s2.display(22,"siva");
         System.out.println("third method");
        s3.display(33,"ram",25);
}
}
```

```
first method
Rno 11
Name      sai
second method
Rno 22
Name      siva
third method
Rno 33
Name      ram
Age 25
```

Constructor

Constructor

- Constructor is a special type of method that is used to initialize the object.
- When an object is created it is automatically invoked.
- It constructs values that is provides data for the object

Rules for creating constructor

There are basically two rules defined for the constructor.

1. Constructor name must be same as its class name. (It has the same as the class name)
2. Constructor must have no explicit return type (It has no return type)

Types of constructors

There are two types of constructors

1. Default constructor (no argument constructor)
2. Parameterized constructor

Constructor-Simple Program

```
//simple program for both constructor
class stu
{
        stu()
        {
                System.out.println("default constructor");
        }
        stu(int a)
        {
                System.out.println("parameterized constructor");
        }
}
public static void main(String a[])
{
    stu s1=new stu();     // default constructor invoked
    stu s2=new stu(10); // parameterized constructor
}
}
```

Output

default constructor
parameterized constructor

Default Constructor

- If there is no constructor in class, the compiler automatically creates the default constructor.
- Examples of default constructor that displays the default values.

```
public class student
{
int rno;
String name;
void display()
{
        System.out.println("Rno        "+rno);
        System.out.println("Name       "+name);
}
public static void main(String a[])
{
         student s1=new student();
         student s2=new student();
          s1.display();
          s2.display();
}   }
```

Output

```
Rno 0
Name     null
Rno 0
Name     null
```

```
public class student
{
int rno;
String name;
student()
{
        rno=0;
        name=null;
}
void display()
{
        System.out.println("Rno        "+rno);
        System.out.println("Name
    "+name);
}
public static void main(String a[])
{
        student s1=new student();
        student s2=new student();
        s1.display();
        s2.display();
}   }
```

Parameterized Constructor

Parameterized Constructor

A constructor that have parameters is known as parameterized constructor.

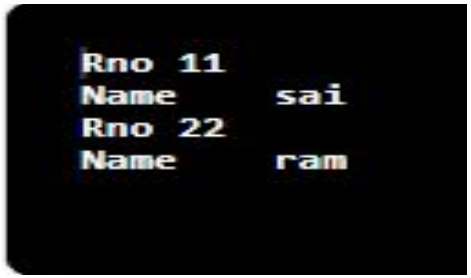
Syntax of default constructor

```
class name(parameter list)
{
}
```

Example

```
class sairam
{
        sairam(parameter list)
        {
        }
}
```

Output



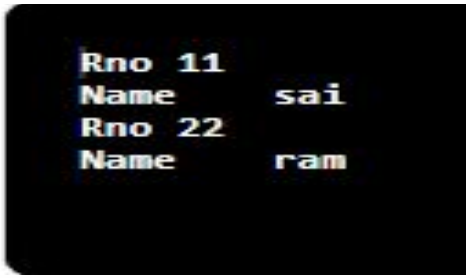

```
public class student
{
int rno;
String name;
student(int r, String n)
{
        rno=r;
        name=n;
}
void display()
{
        System.out.println("Rno        "+rno);
        System.out.println("Name       "+name);
}
public static void main(String a[])
{
        student s1=new student(11,"sai");
        student s2=new student(22,"ram");
        s1.display();
        s2.display();
}  }
```

Constructor Overloading

Constructor Overloading

A constructor overloading is a technique in which a class can have any number of constructors that differ in parameter list.

The compiler differentiates these constructors by taking into account the number of parameters in the list and other types.




```
class stu
{
stu()
{
    System.out.println("default constructor");
}
stu(int a)
{
    System.out.println("constructor with single parameter");
}
stu(int a,int b)
{
    System.out.println("constructor with two parameter");
}
stu(int a, int  b, int c)
{
    System.out.println("constructor with three parameter");
}
public static void main(String a[])
{
    stu s1=new stu();          // default constructor
    stu s2=new stu(1);         // single
    stu s3=new stu(1,2);       // two
    stu s4=new stu(1,2,3);     // three
}    }
```

Video Link

https://www.youtube.com/watch?v=G1Iln3PSrUg