Class ,objects and methods

Syntax to declare class:

```
class class_name
{
   datatype var1;
    ..
   datatype var_n;

    return type method_1(arg list)
    {
     Body of method_1
    }
    …..
    return type method_n(arg list)
    {
    Body of methodn
    }
}
```

Method/member function

- Declaration / prototype
- Definition
- Call
- Various ways of writing function
- 1. Function without arguments
- 2.Function with arguments
- 3.Fumction with argument and return type
- 4.Function without argument and with return type

Syntax of function

- Declaration->

  Returntype functionname(argumentlist);

- Definition->
- Returntype functionname(argumentlist)
- {
- Body of function;
- }
- Function call->
- Functionname();
- Object.Functionname();

- Example
- void add();
- void add(int x, int y);
- ==========
- With return types
- (instead of int it can b double, float, char, string)
- int add(int a, int b);
- int add();

Syntax to declare object:

class name obj_name;

obj_name= new classname();

Ex: Student s;

s=new Student(); or

Student s=new Student;

Syntax to access class members:

Object.classmember

<member can be instance variable or method >

(Ex: instance variable =>s.roll_no,

method=> s.get()

```
class Student
{
int roll_no;
String name;

void get()
{
roll_no=10;
name =“abc”;
}

void put()
{
System.out.println(“Roll no.”+roll_no);
System.out.println(“Name”+name);
}

}
```

```
class  Demo
{
public static void main(String args[])
{
Student s;
s=new Student();
s.get();
s.put();
}
}
```

Program by passing values to methods

```
class Student
{
int roll_no;
String name;

void get (int rn, String n)
{
roll_no= rn;
name = n;
}

void put()
{
System.out.println("Roll no."+roll_no);
System.out.println("Name"+name);
}

}
```

```
class  Demo
{
public static void main(String args[])
{
Student s;
s=new Student();
s.get(101, "sita");
s.put();
}
}
```

Constructor

- S**pecial method** that is used to initialize objects.
- Called when an object of a class is created.
- It can be used to set initial values for object attributes:
- Every class has a constructor either implicitly or explicitly.
- If we don't declare a constructor in the class then JVM builds a default constructor for that class. This is known as **default constructor**.
- A constructor has **same name as the class name** in which it is declared.
- **Constructor must have no explicit return type**.
- Constructor in Java **can not** be **abstract, static, final or synchronized**.
- These modifiers are not allowed for constructor.

syntax

- className (parameters)
- {
- code-statements
- }

Constructor

- Types of constructor

1. Default Constructor- Constructor with empty argument list is called default constructor.
2. Parameterized Constructor –Constructor with argument list is called parameterized constructor.
3. Car c = new Car() //Default constructor
4. Car c = new Car(name); //Parameterized

Program using Default Constructor

```
class Student
{
int roll_no;
String name;

Student()
{
roll_no=10;
name =“abc”;
}

void put()
{
System.out.println(“Roll no.”+roll_no);
System.out.println(“Name”+name);
}

}
```

```
class  Demo
{
public static void main(String args[])
{
Student s=new Student();
s.put();
}
}
```

Program using Parameterized Constructor

```
class Student
{
int roll_no;
String name;

Student(int r ,String n)
{
roll_no=r;
name =n;
}

void put()
{
System.out.println(“Roll no.”+roll_no);
System.out.println(“Name”+name);
}

}
```

```
class  Demo
{
public static void main(String args[])
{
Student s=new Student(10,”abc”);
s.put();
}
}
```

Method overloading

- Method overloading means having two or more methods with the same name but different argument list.

Program using Method Overloading

```
class Demo
{
void test()
{
System.out.println(“hello”);
}

void test(int a)
{
System.out.println(“a=”+a);
}

void test(int a,int b)
{
System.out.println(“a=”+a);
System.out.println(“b=”+b);
}
}
```

```
class  Demo1

{
public static void main(String args[])
{
Demo  d=new Demo();
//function call
d.test();
d.test(10);
d.test(20,30);
}
}
```

Program using Method Overloading

```
class Demo
{
void area(double r)
{
double a;
a=3.14*r*r;
System.out.println(“area of circle”+a);

}
void area(double  len ,double b)
{
double a;
a=0.5*len*b;
System.out.println(“area of
triangle”+a);

}
}
```

```
class  Demo1
{
public static void main(String args[])
{
Demo  d=new Demo();
d.area(3.4);
d.area(2.4,2.6);
}
}
```

Constructor overloading

- Constructor overloading means having two or more constructors with different argument list.

Program using constructor Overloading

```
class Test
{
Test()
{
System.out.println(“hello”);
}

 Test(int a)
{
System.out.println(“a=”+a);
}

 Test(int a,int b)
{
System.out.println(“a=”+a);
System.out.println(“b=”+b);
}
}
```

```
class  Demo
{
public static void main(String args[])
{
Test  t1=new Test();
Test  t2=new Test(10);
Test  t3=new Test(30,40);

}
}
```

Program using Constructor Overloading

```
class Area
{
Area(double r)
{
double a;
a=3.14*r*r;
System.out.println(“area of circle”+a);

}
 Area(double l,double b)
{
double a;
a=0.5*l*b;
System.out.println(“area of
triangle”+a);

}
}
```

```
class  Demo1
{
public static void main(String args[])
{
Area a1=new Area(3.4);
Area a2=new Area(3.4,2.4);

}
}
```