Classes & Objects

Lecture 6

Prof. Anita Agrawal

BITS-Pilani, K.K. Birla Goa campus

Classes

- A *class* is the blueprint from which objects are created
- It is an entity that determines how an object will behave and what the object will contain.
- It defines the variables and the methods common to all objects of a certain kind.
- *Syntax:*

```
class classname
{
    members of class
}
```

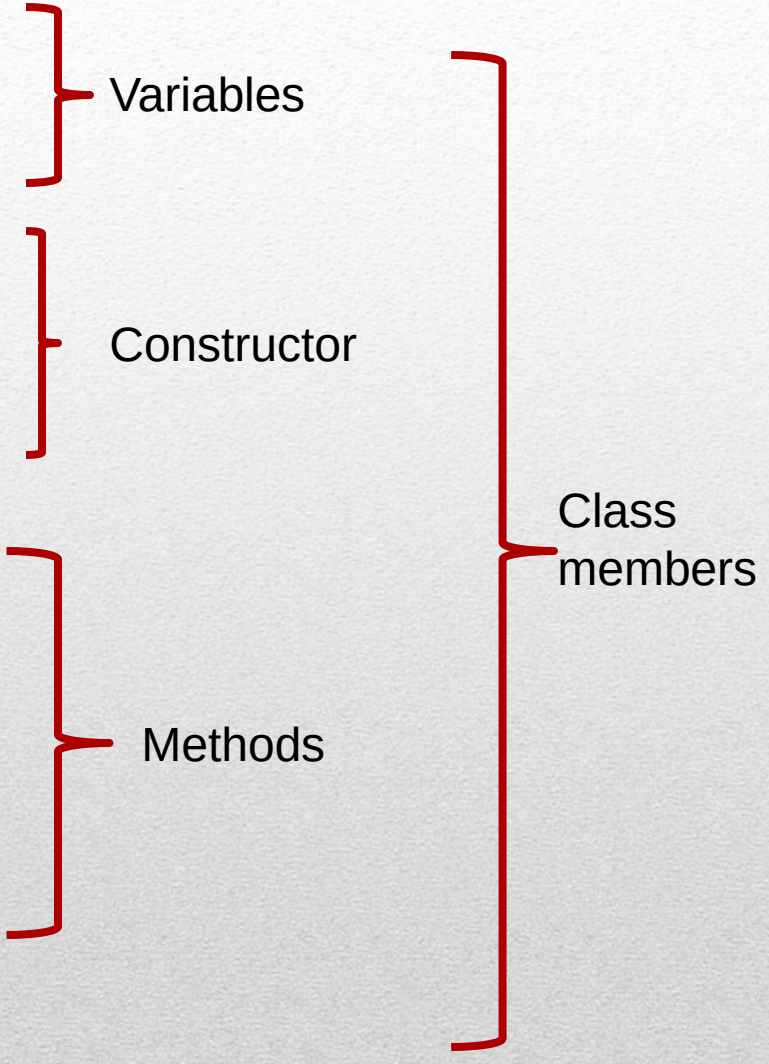
Class members

- Members of class are:
  - Variables
  - Methods
  - Constructors
- Variables: represent the state of the class.
- Methods: represent behavior of the class.
- Constructors: initialise the state of a new instance of the class

```
modifier class clsName
{
 // instance variable declaration
  type1 varName1 = value1;
  :
  typeN varNameN = valueN;

 //  Constructor
  clsName(cparam1)
  {
    // body of constructor
   }


// Methods
rType1 methodName1(mParams1)
 {
   // body of method
  }
  :
rTypeN methodNameN(mParamsN)
  {
   // body of method
   }
}
```



Modifier/ access level

- **Class** access level:
  - Default (accessible only inside the class)
  - Public (accessible in code in any package)
- **Access Level of members: (fields and methods)**
  - Private (only in the class)
  - Package or default (only in this package)
  - Protected (in this package and all sub-classes
  - Public(everywhere)

Packages

- A package is a namespace that organizes a set of related classes and interfaces.

Constructor

- Used for initializing new objects.
- Initializes an object immediately upon creation
- Has the same name as the class in which it resides
- Has optional parameter lists
- Once defined, the constructor is automatically called immediately after the object is created, before the **new** operator completes.
- Has no return type, not even **void**.

```
class rect
 {
     int rlength,rbreadth;
      rect()    //this is a constructor
     {
        rlength=10;
        rbreadth=20;
     }
}
```

Constructor Example

Methods

```
modifier class clsName
{
// instance variable declaration
type1 varName1 = value1;
:
typeN varNameN = valueN;


// Methods
rType1 methodName1(mParams1)
{
// body of method
}
:
rTypeN methodNameN(mParamsN)
{
// body of method
}
}
```

Data: instance variables

code: methods

Methods

- Named methodName1 through methodNameN.
- The return type are rtype1 through rTypeN and
- mParamN are optional parameter lists.

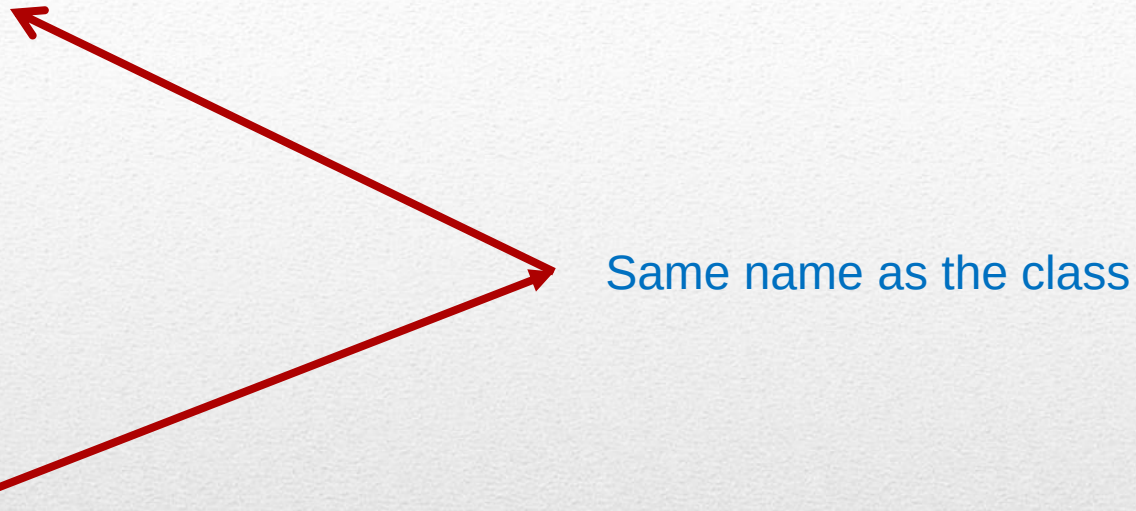
Method Example:

```
int fun_area()
 {
return rlength*rbreadth;
}
```

```
class clsSample
{
   // Variables
     int ctr;
    int i;
    int j;

  // Constructor
    clsSample()
     {
        ctr = 0;
         i = 0;
         j = 0;
      }

// Methods
```



Same name as the class

```
int addition()
  {
    ctr++;
    return i + j;
   }

int NumberofInstances()
   {
      return ctr;
    }
}
```

Objects

- Specimen of the class, a self-contained component consisting of
  - Methods and
  - State
- It determines the behavior of the class.

Objects

- Obtaining objects: two step process.
  - Declare a variable of the class type.
    - does not define an object.
    - simply r*efers* to an object.
  - Acquire an actual, physical copy of the object and assign it to that variable.
    - use the **new** operator.
    - The **new** operator dynamically allocates (that is, allocates at run time) memory for an object and
    - Returns a reference to it.

Syntax

Step 1:

classname object_name

Step 2:

object_name = new classname()

Example:


```
Dog dog1;                 //dog1 contains null,
dog1 = new Dog();  // allocates an actual object
```

Shorthand:

Dog dog1= new Dog();

Each object of class will have its own copy of instance variable: breed, size, color, age

- To access these variables (and methods):
  - Dot(.) operator
- For example:

```
dog1.breed=" german_shepherd ";
dog1.size="Small";
```

OBJECT AND CLASS EXAMPLE: MAIN INSIDE CLASS

```
//Class Declaration
public class Dog
 {
    // Instance Variables
    String breed;
    String size;
    int age;
    String color;

    // method 1
    public String getInfo()
   {
        return ("Breed is: "+breed+" Size is:"+size+" Age is:"+age+" color
is: "+color);
    }

 public static void main(String[ ] args)
 {
        Dog dog1 = new Dog();
       dog1.breed=“german_shepherd";
        dog1.size="Small";
        dog1.age=2;
        dog1.color=“brown";
        System.out.println(dog1.getInfo());
    }
}
```

- Output:

Breed is: german_shepherd Size is:Small Age is:2
color is: brown

Object and Class Example:

```
// Class Declaration
class Dog {
    // Instance Variables
    String breed;
    String size;
    int age;
    String color;

      // method 1
    public String getInfo()
 {
        return ("Breed is: "+breed+" Size is:"+size+" Age is:"+age+" color is:
"+color);
    }
}
public class Execute{
    public static void main(String[] args) {
        Dog dog1 = new Dog();
        dog1.breed=“German_Shepherd";
        dog1.size="Small";
        dog1.age=2;
        dog1.color="white";
        System.out.println (dog1.getInfo());
    }
```

Benefits of Objects

- **Modularity:** The source code for an object can be written and maintained independently of the source code for other objects. Once created, an object can be easily passed around inside the system.
- **Information-hiding:** By interacting only with an object's methods, the details of its internal implementation remain hidden from the outside world.
- **Code re-use:** If an object already exists (perhaps written by another software developer), you can use that object in your program. This allows specialists to implement/test/debug complex, task-specific objects, which you can then trust to run in your own code.

- **Pluggability and debugging ease:** If a particular object turns out to be problematic, you can simply remove it from your application and plug in a different object as its replacement.

- Write down a java program to create a class maruti_suzuki which contains the following:

(i) Variables:

model,

type,

mileage,

auto_transmission,

number of airbags.

(ii) One Method named **show_info** of return type String to display all the info.

(iii) Three objects ob1, ob2, ob3