



UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING
COMPUTER SCIENCE DEGREE PROGRAMME
SECOND YEAR
SCS 2104 - Programming III
Lab Sheet 2

Classes, Objects, Methods, Constructors

1. Real World concept



Who is this?

Why did you call so?

What is it called?

Why is it not a human?



When there are set of common **features** (or **actions**) which comparably identify from a set of things/living beings, we used to call them a special group or a class.



Is this a human?

But he has a name..

He has black hair...

He is wearing something...

He is smiling...



He has a name too...

But he has a beard. He is not smiling.

Is he a human too??

However do both have horns?

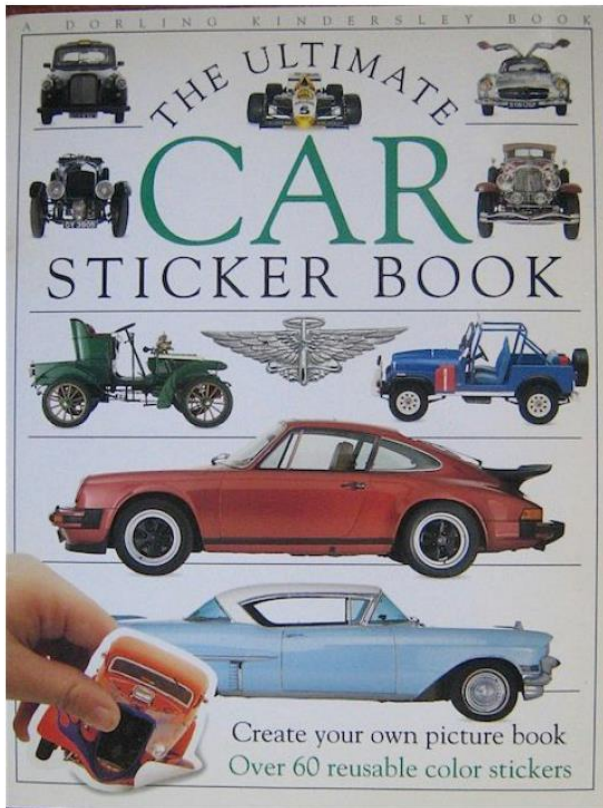
Spot the difference between “class vs. object” and “attribute vs. value”.

Obama is an individual of human class and we can be more specific about him whereas we cannot say exact things about “human”. That means we can exactly say Lincoln has beard, he has a name... etc.

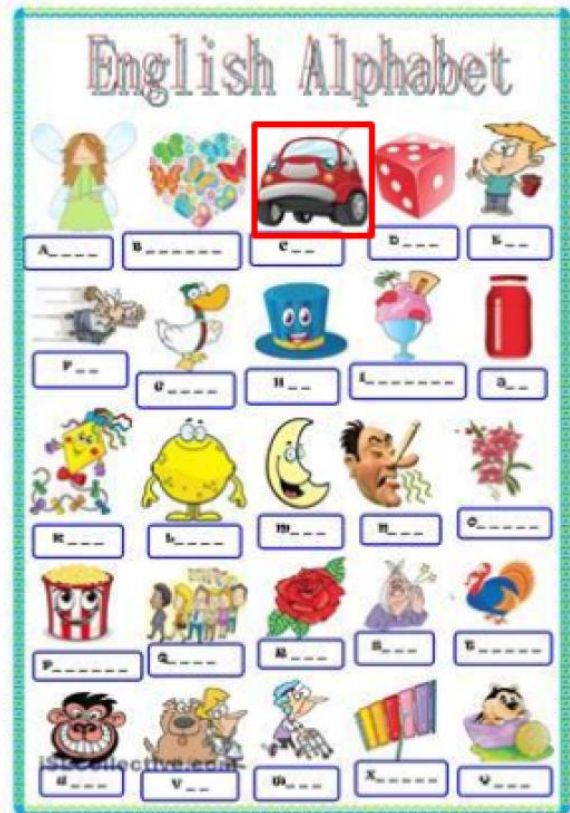
Each individual of same class can have different values for the attributes. But both of them cannot have a feature or an action which does not belong to his or her class (e.g. horns).

Note: Sometimes you may get confused classes and objects.

Consider the group/class name “Car”.



What is “Car” here?




What is “Car” here? a Class?


Another Example:

- Vehicle
 - Car
 - Nissan Car
 - ✓ Leaf
 - ✓ Sunny
 - Toyota Car
 - ✓ Allion
 - ✓ Prius
 - Van
 - Bus

2. Map to Programming

	
Features	Actions
Name:Leader Type:Dobberman Color:Brown Age:3 ..etc	It eats meat. Barks at strangers Runs fast. Does not bite men ..etc

```
Class Tute{  
    Public static void main(String args[]){  
        String name="Leader";  
        String type="Dobberman";  
        String color="Brown";  
        int age=3;  
    }  
}
```

	
Features	Actions
Name:Lassy Type:G.Shepherd Color:Mix Age:5 ..etc	It eats anything. Barks at strangers Runs very fast. Bites men ..etc

```
Class Tute{  
    Public static void main(String args[]){  
        //For Leader  
        String LeaderName="Leader";  
        String LeaderType="Dobberman";  
        String LeaderColor="Brown";  
        int LeaderAge=3;  
  
        //For Lassy  
        String LassyName="Lassy";  
        String LassyType="G.Shepherd";  
        String LassyColor="Mix";  
        int LassyAge=5;  
    }  
}
```

It is still possible to do like that for another 4 or 5 dogs.

But what if,

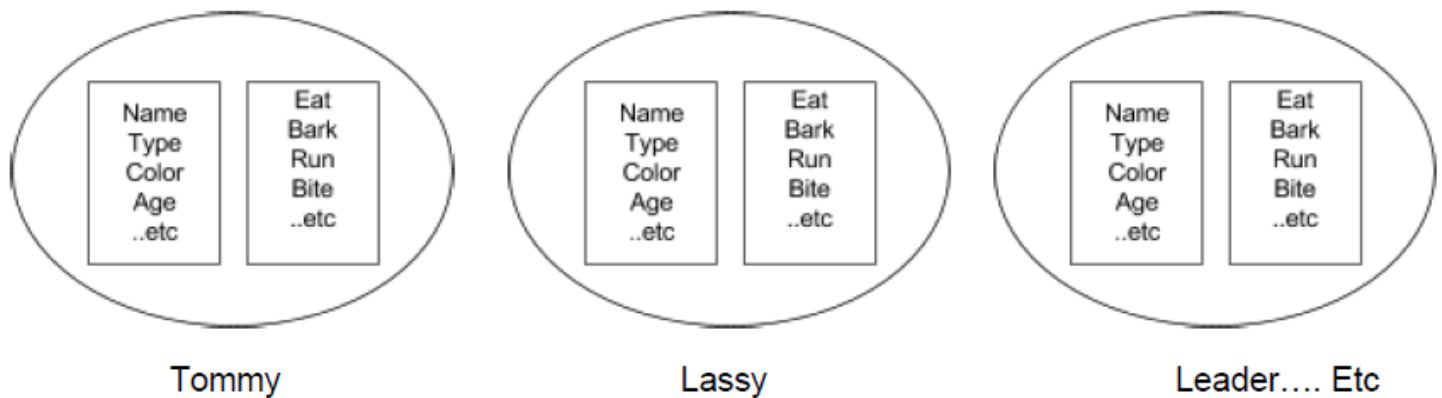
- You came up with 2 Lassys. i.e. Lassy2Type???
- You have to automate a system for 1000s of dogs. Horrible....!!

Okay can't we get the help of array/struct (in C)? Yes you can.

But what about modeling the "Actions". Problem gets worse....

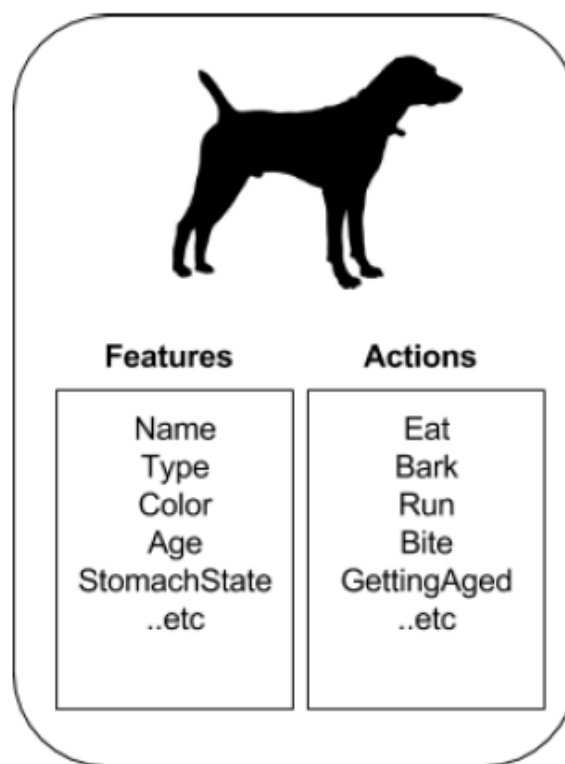
Sometimes it's a disadvantage to directly access data. You'll see how (and why) methods are used to completely hide the data implementation.

However if you recognize the pattern you will be able to generalize the idea...



Blueprint / Class / Template / Contract

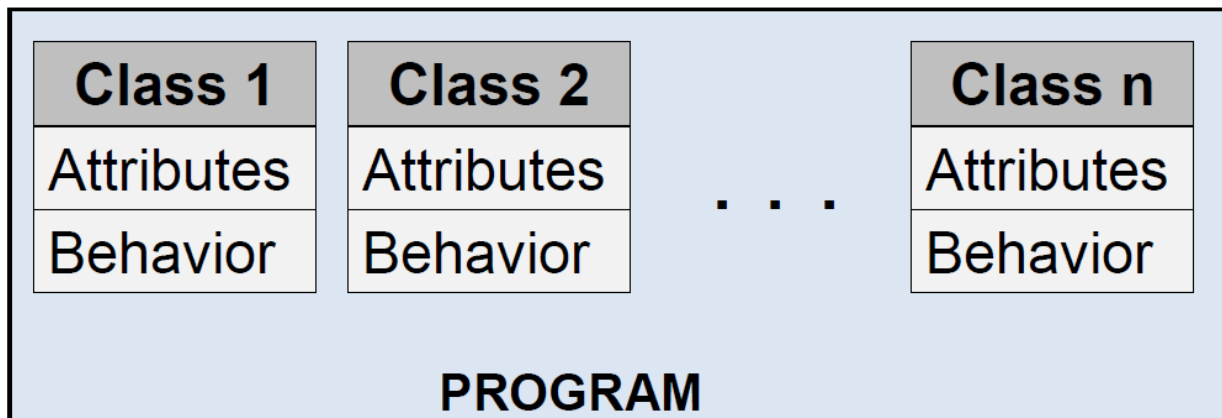
It has,
Features/ States/
Characteristics



Also has,
Behavior/ Action/
Work

3. Classes and Objects

A **class** is a template for multiple objects with similar features. Basic element of Object Oriented Programming is a class. A collection of classes is called a program.



Example 01

Write a Java class to hold the information of students in a class. The basic properties that should be stored are given below.

- Age
- Name of the class (A,B or C)
- Average
- Grade

```
class Student
```

```
{
```

```
    int age;
```

```
    char className;
```

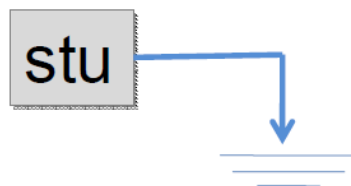
```
    float average;
```

```
    char grade;
```

```
}
```

Instance
Variables

Student stu; ← Reference of Student



- When the reference is created it points to a null value.
- Before access the reference, objects of the class must be declared.
- A class contains a special kind of method called **constructor** whose name is as same as the class and no any return type.

```
class Student
```

```
{
```

```
    Student()
```

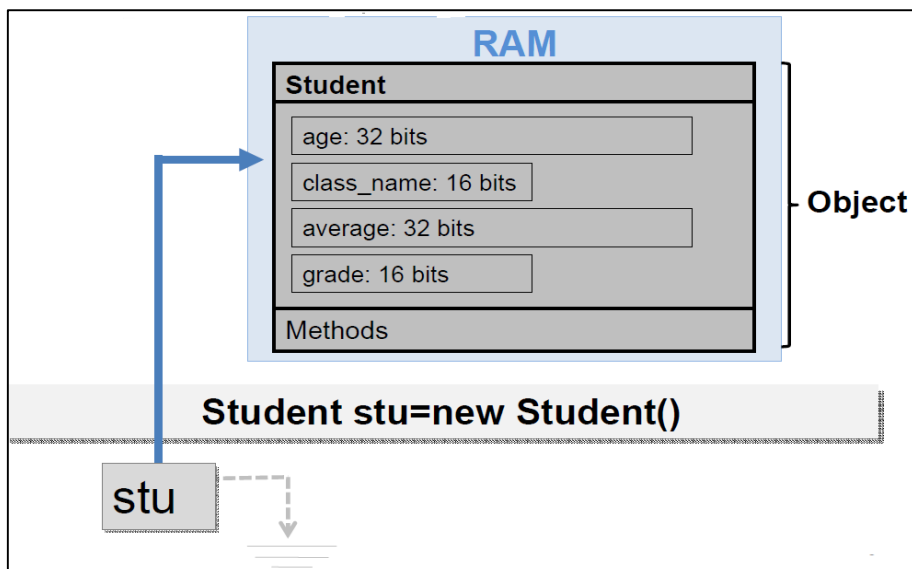
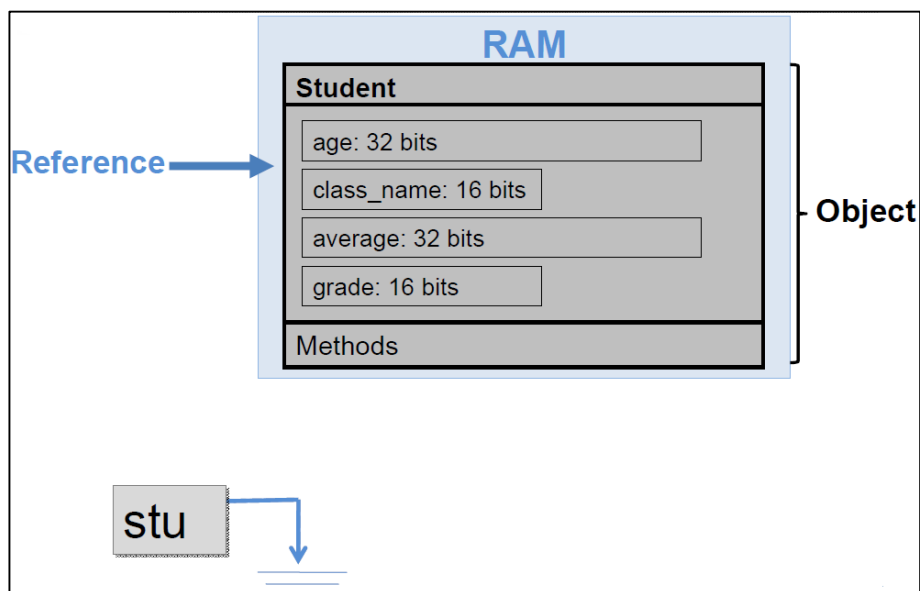
```
    {
```

```
    }
```

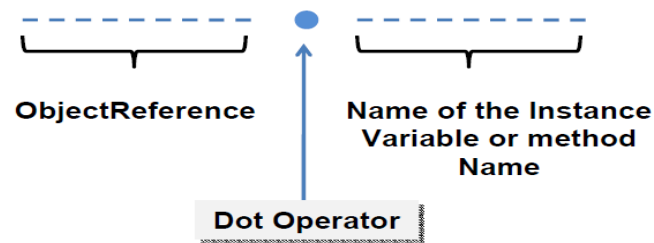
```
}
```

If the programmer does not define any constructors for a class, an empty default constructor is supplied by JAVA.

- Object is an instance of a class. An instance is an individual copy of the class template with its own set of attributes and behaviors.
- Object creation is done with the use of constructor and the **new** operator.
- Once the object is created, a separate memory location is allocated to hold that object.



- The dot operator is used to access the values of the variables and to call the methods of the class.



- Print the value of the instance variables of the class “Student”.

<pre>class Student { int age; char className; float average; char grade; Student() { } }</pre>	<pre>class Example { public static void main(String arg[]) { Student stu=new Student(); System.out.println("Age: "+stu.age); System.out.println("Class Name: "+stu.className); System.out.println("Average: "+ stu.average); } }</pre>
--	--


```
H:\>javac Example.java
H:\>java Example

    Age: 0
    Class Name:
    Average: 0.0

H:\>_
```

Instance Variables have default values.

- Using **Assignment** operator (=) assign the value in the right side of the = operator to the left side.



`stu.age = 10`

Example 02

Write a JAVA program to store the information of two students. Each student have a Age, Gender, Marks for mathematics, Marks for statistics, Marks for computer science and average of three subjects.

Attribute	Student 1	Student 2
Age	21	23
Gender	F	M
Marks for Mathematics	67	89
Marks for Statistics	76	54
Marks for Computer Science	78	95

Step 1: Declare a class to represent student information

```

class Student
{
    byte age;
    char gender;
    byte marks_maths;
    byte marks_stat;
    byte marks_cs;
    float average;
}

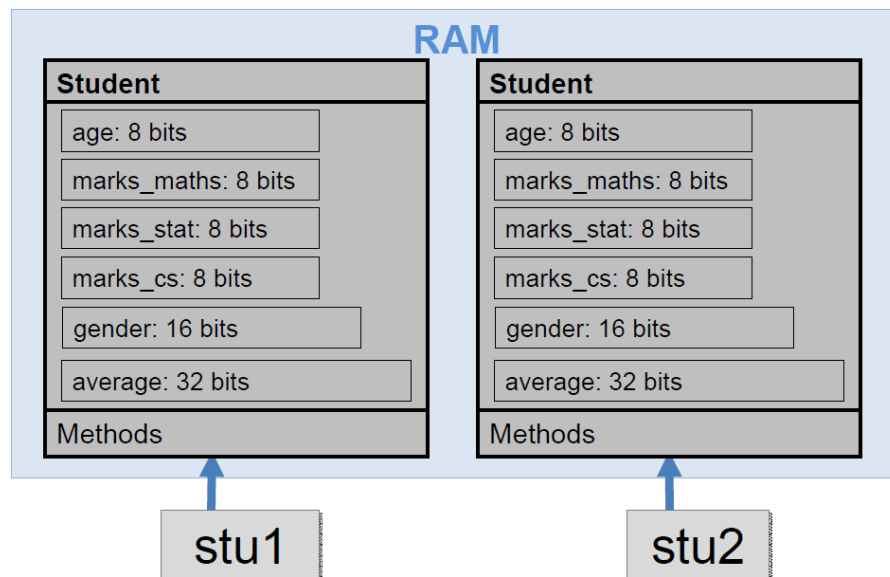
```

Step 2: Define the driver class and the main method.

```
class Example2
{
    public static void main(String arg[])
    {
    }
}
```

Step 3: Create objects from Student class. Since there are two students, we need two objects.

```
class Example2
{
    public static void main(String arg[])
    {
        Student stu1=new Student();
        Student stu2=new Student();
    }
}
```

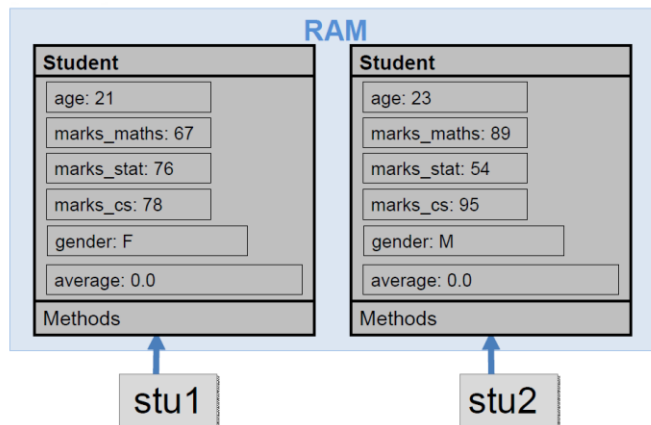


Step 4: Assign the values for attributes.

```
class Example2
{
    public static void main()
    {
        Student stu1=new Student();
        Student stu2=new Student();

        stu1.age=21;

        stu2.age=23;
    }
}
```



Step 5: Print the information of each student.

```
***** STUDENT 1 *****

Age:                21
Mathematics:        67
Statistics:         76
Computer Science:   78
Gender:             F

-----

***** STUDENT 2 *****

Age:                23
Mathematics:        89
Statistics:         54
Computer Science:   95
Gender:             M
```

Step 6: Calculate the average of three subjects for each student.

```
class Example2
{
    public static void main()
    {
        .....
        .....
        .....

        int tot= stu1.marks_maths+stu1.marks_stat+stu1.marks_cs;
        stu1.average=(float)tot/3;
    }
}
```

Note:

The attributes **age**, **gender**, **marks_maths**, **marks_cs**, **marks_stat**, **average** are called as instance variables.

- ❖ The common characteristic mentioned above take different values for each student. Hence they are declared in the class, but outside the methods.

Step 7: Calculate the class average of three subjects.

Class average for one subject does not take different values from one student to another. There is only one value for all the objects created from the same class.

- ❖ If the same value of an attribute is shared by all the class instances are named as **class variables** and are defined as **“static”**. They are declared in the classes, but outside the methods.

Modify the Student class in order to store the class average for three subjects.

```

class Student
{
    byte age;
    char gender;
    byte marks_maths, marks_stat, marks_cs;
    float average;

    static float classAvg_maths;
    static float classAvg_stat;
    static float classAvg_cs;
}

```

Since the class variables are shared by all the instances of the class in which they are defined, they can be accessed directly through the identifier of the class.

```

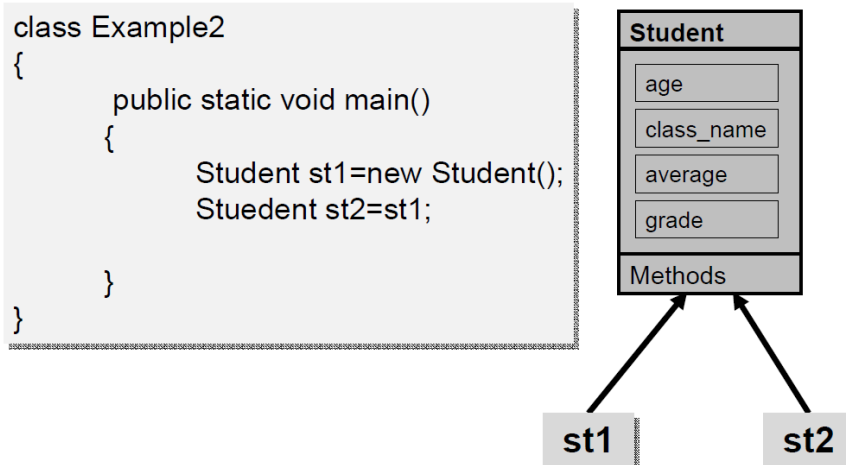
class Example2
{
    public static void main()
    {
        .....
        .....
        .....

        float avg =(float)(stu1.marks_maths+stu2.marks_maths)/2;

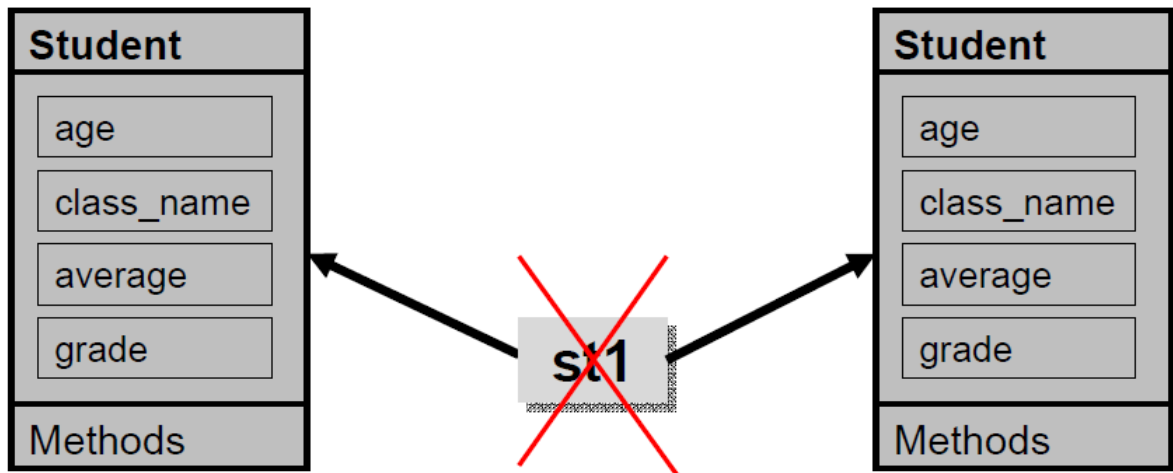
        Student.classAvg_maths=avg;
    }
}

```

Java support for multiple references for same object.



NOTE: One reference can point to only one object for a time. Hence it is impossible to point to multiple objects with the same reference.



- One object could be referenced by multiple references.
- Single reference cannot be pointed to multiple objects.

4. Methods

- Methods are the functions that operate on instances of classes in which they are defined.
- Objects can communicate with each other using methods.
- One class may contain one or more methods.

Method Signature

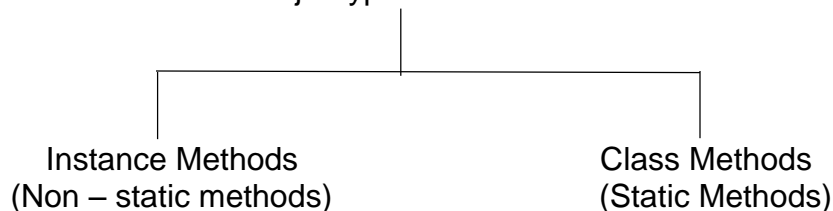
```
return_type method_name (parameter list)
{
    Method_body
}
```

- Method names begin with lower case letters. Do not start with numbers or symbols (@, #, %, etc.).
- If the method name contains more than one word, the starting letter of the second word is capital or they are combined using the underscore (_).
- The symbols like @, #, *, %, etc. are not included in the method names.
- Do not leave spaces in between the words.
- Always use the meaningful names.
- The return type is the primitive data type or object that a method returns.
- If the method does not return any value, then the return type is **void**.
- If the return type is not void, then the returning value should have the same type as the return type.
- The parameter list is a set of variable declarations.
- The list is separated by commas (,).
- The parameters become local variables in the body of the method whose values are passed when the method is called.

```
void calculate(int num, float avg)
{
}

```

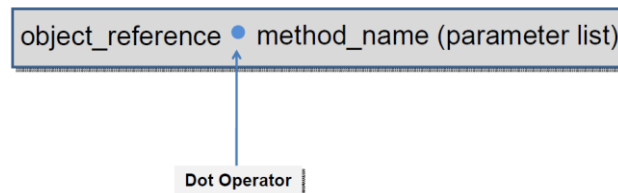
- There are two major types of methods.



Instance Methods

- Instance methods apply and operate on the instances of the class.
- Access Modifiers can be used for instance methods.

- Instance method can access all the instance variables and class variables declared in the same class that the method defines.
- An Instance method can declare any number of local variables which are accessible only by the method in which they defines.
- Instance methods are accessed using the dot operator.



- **'this'** keyword is used inside any instance method to refer to the current object which the current method has been called.
- **this'** keyword is important where a reference to an object of the current class type is required.
- Methods declared with static (class methods) cannot use 'this' keyword.

```

class Student
{
    byte tot;

    void getTotal()
    {
        this.tot=40;
    }
}
  
```

The object 'st' calls the instance method 'getTotal()'.
Hence, 'this' refers to the object 'st'.

```

class Test
{
    public static void main(String arg[])
    {
        Student st=new Student();
        st.getTotal();
    }
}
  
```

From Example 02

Define a method to calculate the average of three subjects for each student.

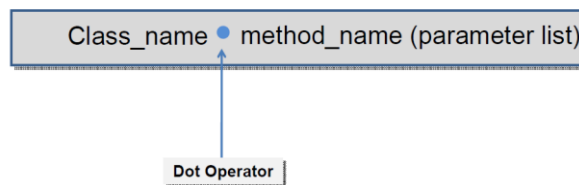
```

class Student
{
    byte age;
    char gender;
    byte m_maths, m_stat, m_cs;
    byte average;
    float avg;

    void calculateAverage()
    {
        this.avg= (float)((this.m_maths+
                           this.m_stat+this.m_cs)/3);
    }
}
  
```

Class / Static Methods

- Class methods operate on the classes and they are defined using the keyword 'static'.
- Access Modifiers can be used for class methods.
- Class methods can access only the class variables declared in the same class that the method defines. They are not allowed to access the instance variables.
- A class method can declare any number of local variables which are accessible only by the method in which they defines.
- Class methods are accessed using the dot operator and the name of the class which defines the method.



Note: main method is also a class method

From Example 02

Define a method to calculate the average of three subjects for each student.

```
Class Average for Mathematics:      78.0
Class Average for Statistics:        65.0
Class Average for Computer Science:  86.5
```

Method Overloading

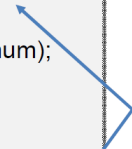
- JAVA allows to have more than one method with the same name, but different method signatures. This difference occurs by means of the parameter list, but not by return type.
- The differences between the parameter lists are obtained based on the number or data types of the parameters passes to the methods.

```

class Example5
{
    void display(int num)
    {
        System.out.println(num);
    }

    void display(float num)
    {
        System.out.println(num);
    }
}

```



Overloaded methods

Constructor Overloading

- As same as the other methods, constructors can also be overloaded, with the same name (name of the class), but different parameter lists.
- Overloaded constructors enable creation of the objects with required properties.

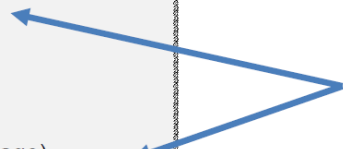
```

class Student
{
    int age;

    Student()
    {
    }

    Student(int age)
    {
        this.age=age;
    }
}

```



Overloaded Constructors

- The required parameters are passed when the constructor is called.

```

class Example
{
    public static void main(String arg[])
    {
        Student st=new Student(23);
    }
}

```

- A constructor can call another constructor using “**this**” keyword.

```
class Student
{
    int age;

    Student()
    {
        System.out.println("Welcome to Student class");
    }

    Student(int age)
    {
        this(); ← The calling statement of another
        this.age=age; constructor should be the first
    }
}
```

Exercises

1. Define a dog class mentioned above and give birth to Leader and Lassy.
 - a. By birth their stomach is empty.
 - b. Feed some meat to Lassy.
 - c. If a strange man comes near by simulate the reaction of Leader.
 - d. One years gone.
 - e. Keep track of siblings.
2. Create a Point.java class and add a functionality to,
 - a. Get the distance from the root
 - b. Move a point and how can we get the distance of two points?
3. Create Number class. Initially it has value 0. It should be able to increment, decrement and get the negation. Create another class to do number operations (for 2 numbers) like addition and multiplication.
4. Create a class for a Die. It should change its value randomly each time it rolls. Extend the program into two dice game. Roll two dice and if the total gets 7, you win.

5. Write a java class called Appointment.java representing an appointment you have with a person. This class should store three bits of information about an appointment:

- With whom the appointment is
- Day of the appointment
- Appointment duration

The Appointment class should have six methods

- setName()
- getName()
- setDay()
- getDay()
- setDuration()
- getDuration()

Let the user create two Appointment objects and then prints appointment details.

Sample output:

Enter appointment one person: Chamara Peris

Enter day of the appointment one: Monday

Enter the duration of appointment one: 1

Enter appointment two person: Kasun Kalhara

Enter day of the appointment two: Tuesday

Enter the duration of appointment two: 1.5

You have a 1 hour appointment on Monday with Chamara Peris.

You have a 1.5 hour appointment on Tuesday with Kasun Kalhara.

6. Write a Java program to define a calculator. The calculator should be able to store two numbers and perform following operations on them.

- Add
- Subtract
- Multiply
- Divide

7. Write a JAVA program to calculate the surface area and the volume of a cube. You must define separate operations to calculate the surface area and the volume.
8. Write a JAVA program to calculate the average of numbers based on following options.
 - Average of two integer numbers
 - Average of three integer numbers
 - Average of an integer number and a float number
 - Average of two integer numbers and a float number
 - Average of three float numbers
 - Average of two float numbers and two integer values
 - Average of two double values
 - Average of three numbers of type integer, double and float

You must use overloaded methods to perform the required operations.