Core Java - 3 (Selenium)

24 May 2017 PM 02:03

Classes and Objects (Back to Top)

Let us consider one more example:

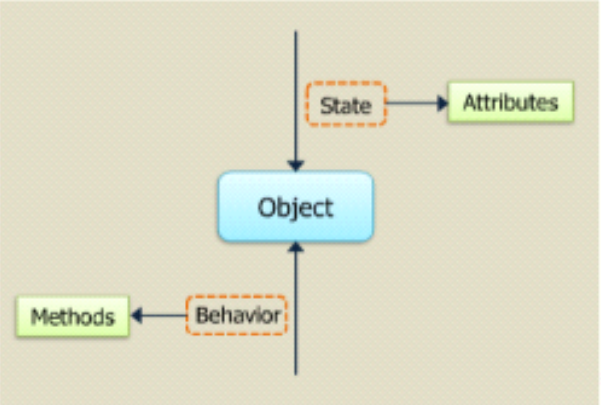
→ When we construct a house, we create a plan. This is called the blueprint of the house. The house is build, based on the plan.

→ Classes and objects are similar to this. Classes are like plan and objects are similar to the constructed house.

Attributes and Methods (Back to Top)

→ Attributes are state of an object.

→ Methods are behaviour of an object.



Classes in Java (Back to Top)

A Class in Java is defined using the keyword "class".

For example:

```
public class student {
    int reg_no;
    String name;
    String stream;

    void Study() {
    }

    void WriteExams() {
    }

    void AttendClasses() {
    }

    void WriteAssignments() {
    }
}
```

Setter / Getter Methods

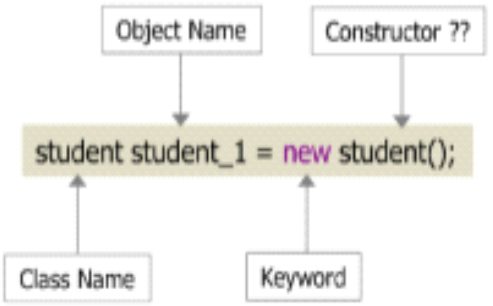
Calling these method to set the values (Later set the values with the help of constructor)

Objects in Java (Back to Top)

Objects to a class is created by using the keyword "new".

new allocates memory for the object.

For example:



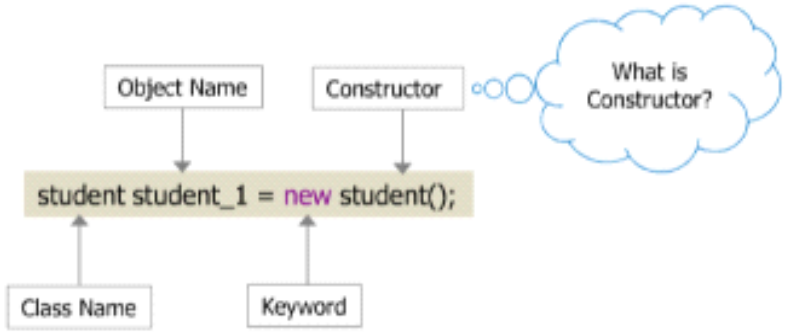
Here student_1 is the object of the class student.
With student_1 any of the method of the class can be invoked.

Constructors (Back to Top)

Objects to a class is created by using the keyword "new".

new allocates memory for the object.

For example:



Here student_1 is the object of the class student.
With student_1 any of the method of the class can be invoked.

Why do we use Constructors?

Classes will have attributes. Before using them, if they need to be initialized then it has to be placed in a method and this method has to be called every time the object is being created.

To avoid this process, a special function by the same name of the class is created for initializing.

Where do we use Constructors?

Constructors are used in all the cases where objects are being created. They are called wherever the class attributes requires initialization.

A constructor:

- → is used in the creation of an object.
- → is a special method with no return type.
- → must have the same name as the class it is in.
- → is used to initialize the object.
- → if not defined, will initialize variables to default value.

Incase a constructor is not defined then a default constructor is called which initializes the instance variables to default value.

The moment object of a class is created, constructor of the class is called which initializes the class attributes.

Default Constructor and Parameterized constructors (Back to Top)

Constructor Overloading (Back to Top)

Encapsulation in Java (After Access Modifiers)

Encapsulation in Java is a process of wrapping code and data together into a single unit, for example, a capsule which is mixed of several medicines.

Capsule

Advantage of Encapsulation in Java

By providing only a setter or getter method, you can make the class read-only or write-only. In other words, you can skip the getter or setter methods.

It provides you the control over the data. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to

store the negative numbers in the setter methods.

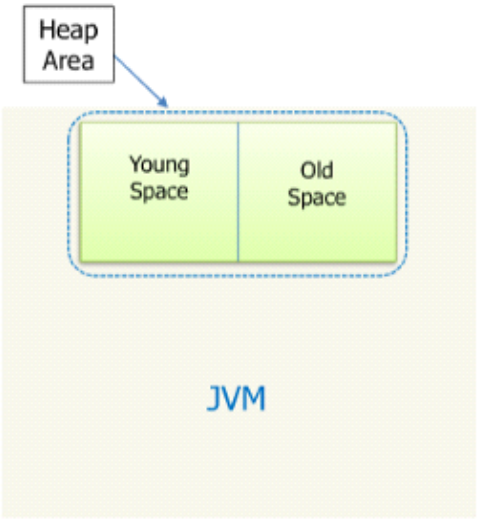
It is a way to achieve data hiding in Java because other class will not be able to access the data through the private data members.

Memory Management in Java (Back to Top)

Heap and Garbage collector (Back to Top)

- → The memory area in JVM where objects are created is called Heap.
- → Heap is divided into two parts. Young space and old space.
- → The memory is freed during runtime by a special thread called Garbage Collector.
- → The Garbage Collector looks for objects which are no longer needed by the program and destroys them.
- → All the newly allocated objects are created in young space. Once the young space is full then garbage collector is called so that memory can be released.
- → If the object has lived for long in young space then they will be moved to old space. Once the old space is full, garbage collector is called to release the space in heap.

Heap - inside JVM (Back to Top)



Inheritance (Back to Top)

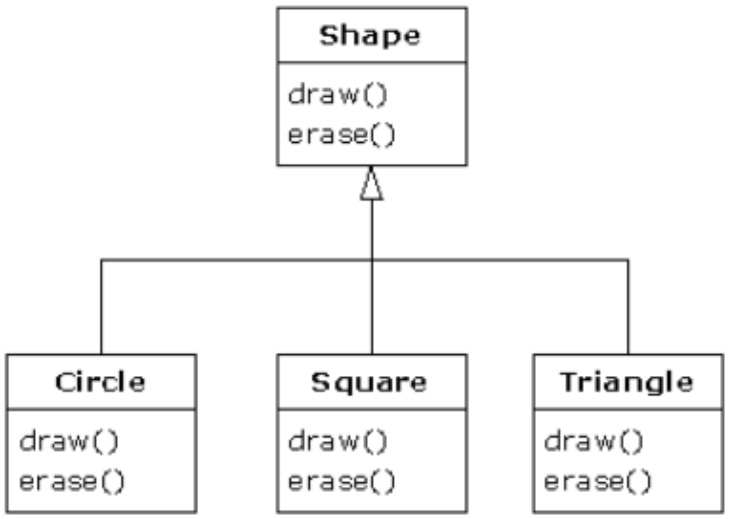
Why do we use inheritance?

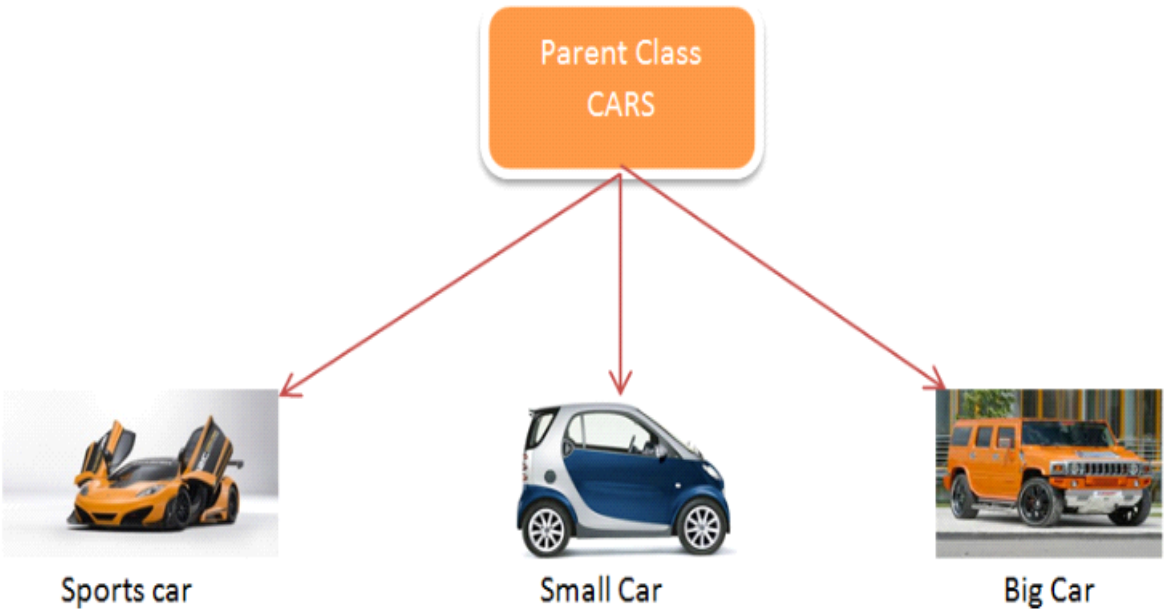
Since some common properties and methods are required by few classes hence they have to be implemented in all those classes.

To avoid code redundancy, a class is developed with the common attributes and methods and it is used as a base class for the derived classes.

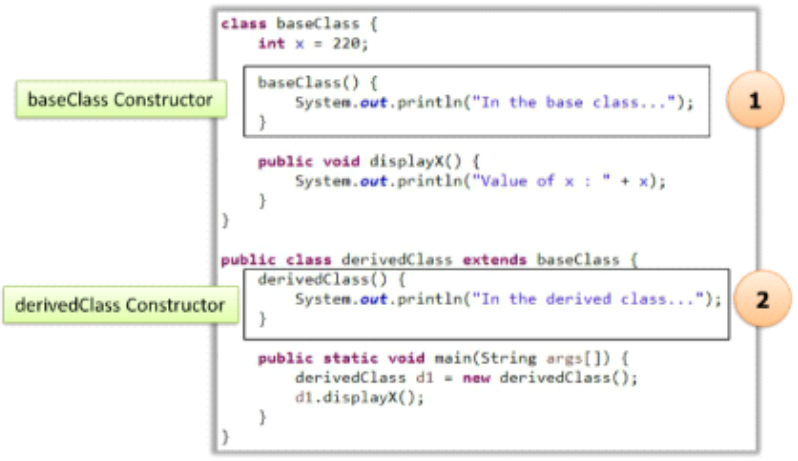
Where do we use inheritance?

Inheritance can be used in all places where parental properties and methods are used.



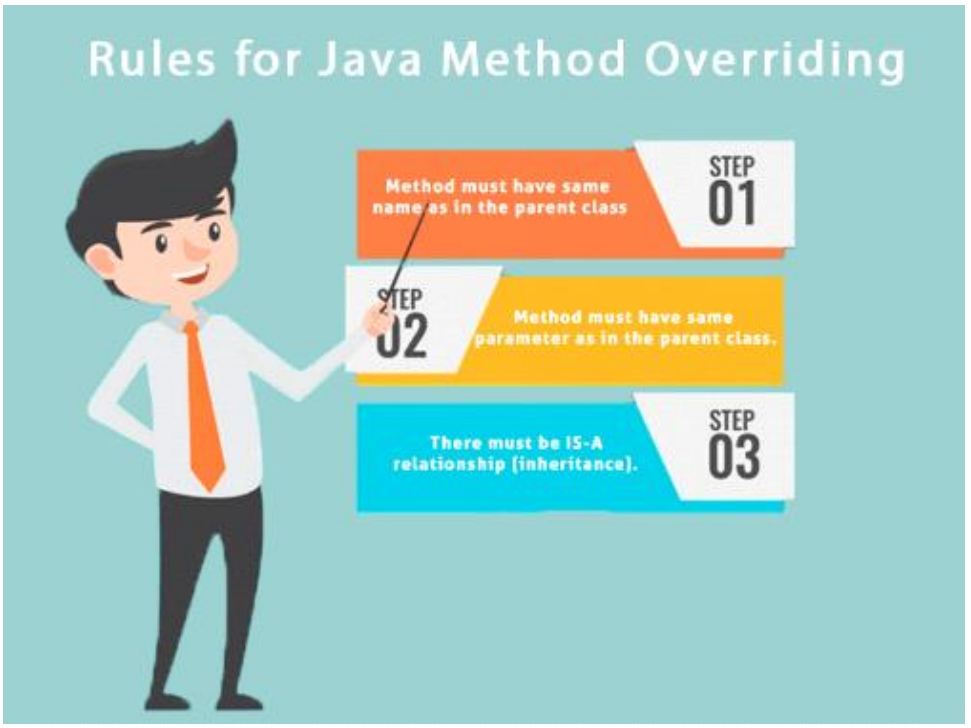


Constructor Chaining in Inheritance (Back to Top)




```
class baseClass {
    int x = 220;

    baseClass() {
        System.out.println("In the base class...");
    }

    public void displayX() {
        System.out.println("Value of x : " + x);
    }
}

public class derivedClass extends baseClass {
    derivedClass() {
        System.out.println("In the derived class...");
    }

    public static void main(String args[]) {
        derivedClass d1 = new derivedClass();
        d1.displayX();
    }
}
```

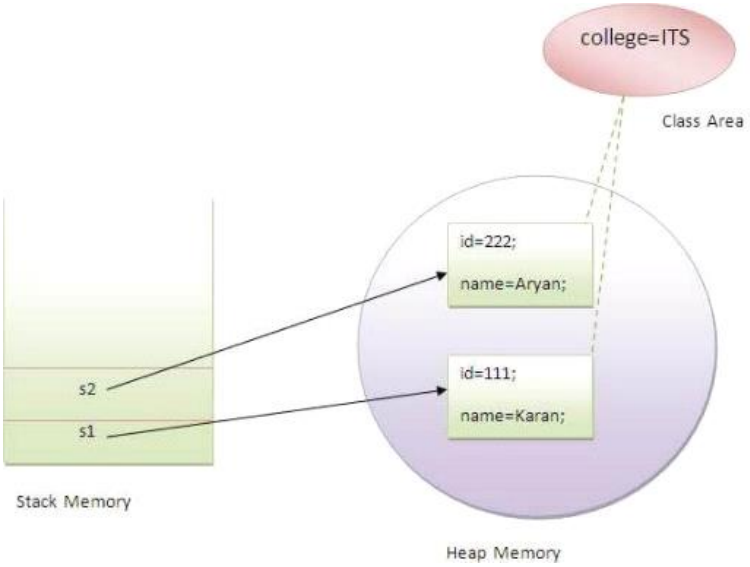
Method Overriding / Runtime Polymorphism (Back to Top)



Super keyword

Final variable, method, class (Back to Top)

Static (Back to Top)



Abstraction in Java

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

Ways to achieve Abstraction
There are two ways to achieve abstraction in java

Abstract class (0 to 100%)
Interface (100%)

Abstract Classes (Back to Top)