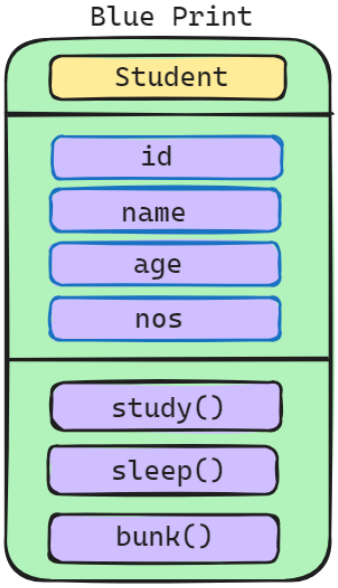


Class Name	Student
Attributes	id, name, age, nos
Behaviours	study, sleep, bunk

How to create a class and object in Java

1. Must define some blue print - class

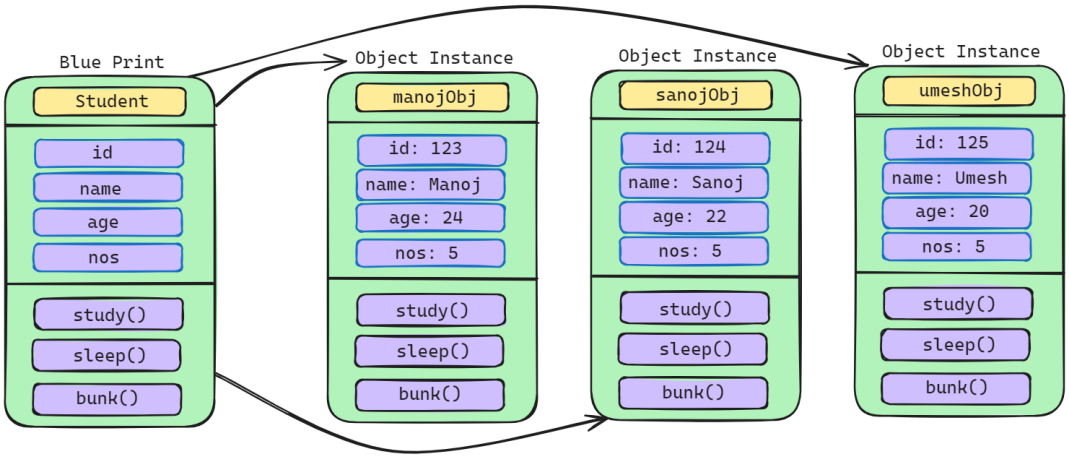
```
public class Student {
    // Attributes (Variables)
    public int id;
    public String name;
    public int age;
    public int nos;

    // Behaviours (Methods)
    public void study(){
        System.out.println(name + "studying");
    }
    public void sleep(){
        System.out.println(name + "sleeping");
    }
    public void bunk(){
        System.out.println(name + "bunking");
    }
}
```

- Class is a description of an object's `property` and `behaviour`
- Creating class is as good as creating `data type`
- Class is defining a category of new data

2. Object an instance of class

```
public class App {
    public static void main(String[] args) throws Exception {
        Student manojObj = new Student();
        manojObj.id = 123;
        manojObj.name = "Manoj";
        manojObj.age = 24;
        manojObj.nos = 5;
        manojObj.study();

        Student sanojObj = new Student();
        sanojObj.id = 124;
        sanojObj.name = "Sanoj";
        sanojObj.age = 22;
        sanojObj.nos = 5;
        sanojObj.sleep();

        Student umeshObj = new Student();
        umeshObj.id = 125;
        umeshObj.name = "Umesh";
        umeshObj.age = 20;
        umeshObj.nos = 5;
        umeshObj.bunk();
    }
}

/*
Expected Output:
Manoj studying
Sanoj sleeping
Umesh bunking
*/
```

- Object is a real world entity
- Object is an instance of a class
- Object `consumes memory` to hold property values



3. What is constructor?

- Constructor is a member function of a class
- The name of constructor is same as the name of the class.
- Constructor has no return type.

📝Constructor is special:

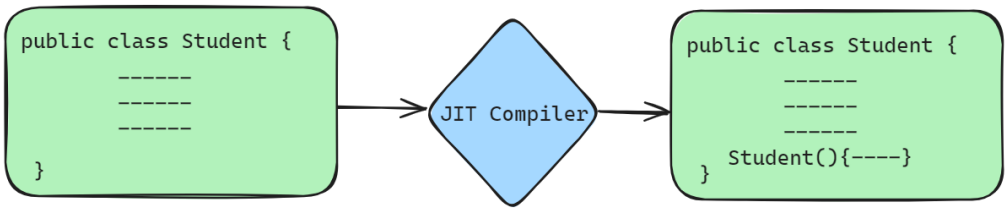
- A constructor is a special method that is `used to initialize a newly created object` and is called `implicitly`, just after the memory is allocated for the object.
- It is not mandatory for the coder to write a constructor for the class.
- When there is no constructor defined in the class by the programmer, the compiler implicitly provides a `default constructor` for the class.

4. Default Constructor Example

```
public class Student {
    // Attributes (Variables)
    public int id;
    public String name;
    public int age;
    public int nos;

    // Default constructor
    public Student(){
        System.out.println("Student default CTOR is called!!");
    }

    // Behaviours (Methods)
    public void study(){
        System.out.println(name + " studying");
    }
    public void sleep(){
        System.out.println(name + " sleeping");
```

```
    }
    public void bunk(){
        System.out.println(name + " bunking");
    }
}
public class App {
    public static void main(String[] args) throws Exception {
        Student manojObj = new Student();
        manojObj.id = 123;
        manojObj.name = "Manoj";
        manojObj.age = 24;
        manojObj.nos = 5;
        manojObj.study();

        Student sanojObj = new Student();
        sanojObj.id = 124;
        sanojObj.name = "Sanoj";
        sanojObj.age = 22;
        sanojObj.nos = 5;
        sanojObj.sleep();

        Student umeshObj = new Student();
        umeshObj.id = 125;
        umeshObj.name = "Umesh";
        umeshObj.age = 20;
        umeshObj.nos = 5;
        umeshObj.bunk();
    }
}

/*
Expected Output:
Student default CTOR is called!!
Manoj studying
Student default CTOR is called!!
Sanoj sleeping
Student default CTOR is called!!
Umesh bunking
*/
```

Rule: If there is no constructor in a class, compiler automatically creates a default constructor.



5. Parameterized Constructor example

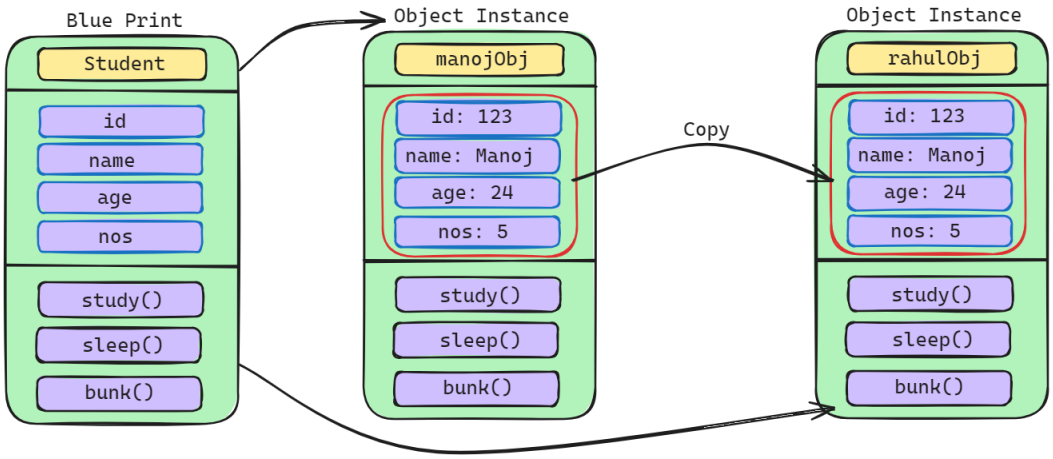
```
public class Student {
    // Attributes (Variables)
    public int id;
    public String name;
```

```
    public int age;
    public int nos;

    // Parameterized constructor
    public Student(int id, String name, int age, int nos){
        System.out.println("Student parameterized CTOR is called!!");
        // Initialize the memeber variables
        this.id = id;
        this.name = name;
        this.age = age;
        this.nos = nos;
    }

    // Behaviours (Methods)
    public void study(){
        System.out.println(name + " studying");
    }
    public void sleep(){
        System.out.println(name + " sleeping");
    }
    public void bunk(){
        System.out.println(name + " bunking");
    }
}
public class App {
    public static void main(String[] args) throws Exception {
        Student manojObj = new Student(123, "Manoj", 24, 5);
        manojObj.study();

        Student sanojObj = new Student(124, "Sanoj",22,5);
        sanojObj.sleep();

        Student umeshObj = new Student(125, "Umseh", 20, 5);
        umeshObj.bunk();
    }
}

/*
Expected Output:
Student parameterized CTOR is called!!
Manoj studying
Student parameterized CTOR is called!!
Sanoj sleeping
Student parameterized CTOR is called!!
Umseh bunking
*/
```

6. Copy Constructor Example

```
public class Student {
    // Attributes (Variables)
    public int id;
    public String name;
    public int age;
    public int nos;

    // Parameterized constructor
    public Student(int id, String name, int age, int nos){
        System.out.println("Student parameterized CTOR is called!!");
        // Initialize the memeber variables
```

```
        this.id = id;
        this.name = name;
        this.age = age;
        this.nos = nos;
    }

    // Copy constructor
    // sourceObj: it represent the manojObj
    public Student(Student sourceObj){
        System.out.println("Student copy CTOR is called!!");
        // Initialize the member variable through manojObj
        this.id = sourceObj.id;
        this.name = sourceObj.name;
        this.age = sourceObj.age;
        this.nos = sourceObj.nos;
    }

    // Behaviours (Methods)
    public void study(){
        System.out.println(name + " studying");
    }
    public void sleep(){
        System.out.println(name + " sleeping");
    }
    public void bunk(){
        System.out.println(name + " bunking");
    }
}
public class App {
    public static void main(String[] args) throws Exception {
        // Parameterized CTOR
        Student manojObj = new Student(123, "Manoj", 24, 5);
        manojObj.study();

        // Copy CTOR
        Student rahulObj = new Student(manojObj);
        rahulObj.study();
    }
}
/*
Expected Output:
Student parameterized CTOR is called!!
Manoj studying
Student copy CTOR is called!!
Manoj studying
*/
```



7. Object life cycle

In Java, the life cycle of an object refers to its creation, usage, and eventual destruction by the `garbage collector`. Here are the main stages of an object's life cycle in Java:

🚀`State 1:` **Creation (Instantiation):**

- An object is created using the `new` keyword followed by a constructor.

```
MyClass obj = new MyClass();
```

🚀`State 2:` **Initialization:**

- After creation, the object's fields are initialized, either explicitly through constructor parameters or through default values if no constructor is provided.

🚀`State 3:` **Usage:**

- The object is used by invoking its methods and accessing its fields. It serves its purpose during this phase.

🚀`State 4:` **Reference:**

- Objects may be referenced by variables, fields, or other objects. As long as there is a reference to an object, it remains reachable and won't be eligible for garbage collection.

🚀`State 5:` **Dereference:**

- When a reference to an object is set to `null` or goes out of scope, the object becomes eligible for garbage collection.

🚀`State 6:` **Garbage Collection:**

- The garbage collector in Java is responsible for reclaiming memory occupied by objects that are no longer reachable. The `JVM` automatically identifies and collects these unreferenced objects.

🚀`State 7:` **Finalization (Optional):**

- Before an object is reclaimed by the garbage collector, the `finalize()` method (if overridden) is called. This method can be used for cleanup tasks.

🚀`State 8:` **Destruction:**

- Once an object has been garbage collected, its memory is deallocated, and it no longer exists in the program.