CS6308- Java Programming

V P Jayachitra
Assistant Professor
Department of Computer Technology
MIT Campus
Anna University

MODULE II JAVA OBJECTS -1	L	T	P	EL
	3	0	4	3

Classes and Objects, Constructor, Destructor, Static instances, this, constants, Thinking in Objects, String class, Text I/O

SUGGESTED ACTIVITIES :

- Flipped classroom
- Practical - Implementation of Java programs – using String class, Creating Classes and objects
- EL – Thinking in Objects

SUGGESTED EVALUATION METHODS:

- Assignment problems
- Quizzes

Constructor

```
public class  Sample{
        public Sample(){   //constructor


        }
}
```

```
//Syntax constructor
accessModifier ClassName() {
   // Initialization code
}
```

No Return Type: no return type including void

Implicit Return: implicitly return reference of newly created object

Naming: Constructor name is same as classname

Access Modifiers: can be public , private, protoected

Constructor :Default Constructor

- **A default constructor is a no-argument constructor**
- **Default constructor is provided by the Java compiler if no other constructors are defined in the class**
- **Initializes member variables to default values**

```
public class MyClass {
   int num;
   String str;

   // Default constructor
     public MyClass() {
     }
}
```

Parameterized public constructor

```
public class MyClass {
  int value;


  //Parameterized  Public constructor
  public MyClass(int value) {
    this.value = value;
  }
}

public class Main {
  public static void main(String[] args) {
    MyClass obj = new MyClass(10);
    System.out.println(obj.value);
  }
}
```

```
10
```

- Constructors are intended to initialize instance variables when an object is created.
- Takes arguments that are used to set initial values for object instance variables

Parameterized public constructor

```
public class MyClass {
   int value;

 private static int staticVar; // Static variable

// Static initialization block
 static {
              staticVar = 30; }

   public MyClass(int value) { //Parameterized  Public constructor
      this.value = value;
   }
}
public class Main {
   public static void main(String[] args) {
      MyClass obj = new MyClass(10);
      System.out.println(obj.value);
      System.out.println(MyClass.staticVar);
   }
}
```

```
10
30
```

- Constructors are intended to initialize instance variables when an object is created.
- Constructors cannot directly initialize static variables.
- Takes arguments that are used to set initial values for object instance variables

Protected constructor

```
public class Base {
   int value;
   // Protected constructor
   protected Base() {
      value=10;
   }
}
public class Derived extends Base {
   public Derived() {
      super();            // Calls the protected constructor of Base
   }
}
public class Main {
   public static void main(String[] args) {
      Derived obj = new Derived();
      System.out.println(obj.value);
   }
}
```

```
10
```

- The constructor can be accessed within the same package and by subclasses

Private constructor

```
public class Singleton {
   private static Singleton instance;   //to hold single instance of the class
   // Private constructor
   private Singleton() {

   }
   // Static method to get the singleton instance
   public static Singleton getInstance() {
      if (instance == null) {
         instance = new Singleton();
      }
      return instance;
   }
}
public class Main {
   public static void main(String[] args) {
      Singleton obj = Singleton.getInstance(); // Access via static method
      System.out.println(obj);
   }
}
```

```
Singleton@15db9742
```

- private constructor ensures that no other class can directly create an instance of the class.

Default parametrized constructor

```
public class MyClass {
  int value;

  //Default parametrized constructor: i.e  no access modifier stated
   MyClass(int value) {
    this.value = value;
  }
}
```

10

```
public class Main {
  public static void main(String[] args) {
    MyClass obj = new MyClass(10);   // Valid within the same package
    System.out.println(obj.value);
}
}
```

- Constructors are intended to initialize instance variables when an object is created.
- Takes arguments that are used to set initial values for object instance variables
- .

Default constructor

```
public class MyClass {
  int value;

  //Default parametrized constructor
   MyClass(int value) {
     this.value = value;
  }
}

public class Main {
  public static void main(String[] args) {
    //MyClass obj = new MyClass();  compilation error
    MyClass obj = new MyClass(10);    // Valid within the same package
    System.out.println(obj.value);
}
}
```

10

- Constructors are intended to initialize instance variables when an object is created.
- Takes arguments that are used to set initial values for object instance variables
- Once you define any parameterized constructor in a class, the default no-argument constructor is not automatically provided. If you need a no-argument constructor, you must explicitly define it alongside any parameterized constructors.

Default constructor

```
public class MyClass {
   int value;
   MyClass() {
      this.value = value;
   }

   //Default constructor
    MyClass(int value) {
      this.value = value;
   }
}

public class Main {
   public static void main(String[] args) {
      MyClass obj = new MyClass();
      MyClass obj = new MyClass(10);    // Valid within the same package
      System.out.println(obj.value);
}
}
```

10

- Constructors are intended to initialize instance variables when an object is created.
- Takes arguments that are used to set initial values for object instance variables
- Once you define any parameterized constructor in a class, the default no-argument constructor is not automatically provided. If you need a no-argument constructor, you must explicitly define it alongside any parameterized constructors.

Copy constructor

```
public class MyClass {
   int num;
   public MyClass() {
    }
   // Copy constructor
   public MyClass(MyClass other) {
      this.num = other.num;
   }
   public static void main(String[] args) {
      MyClass original = new MyClass();
      original.num=10;

      MyClass copy = new MyClass(original);

      System.out.println("Original object:");
     System.out.println (original.num)

      System.out.println("Copied object:");
    System.out.println (copy.num)   } }
```

```
Original object:
10
Copied object:
10
```

- A copy constructor is used to create a new object as a copy of an existing object.

this

- ‘this’ is a keyword used as a special reference variable that refers to the current object of the class
- Used to refer current object instance variable within the class.
    - Distinguishes the instance variable from parameters with the same name
    - Avoids ambuigty

```
public class Student {
    private String name,regno; // Instance variables
    // Default constructor
    public Student() { // Initialize with default values
        this.name = "Unknown";
        this.regno = "0000";       }
    public Student(String name, String regno) { // Parameterized constructor
        this.name = name;
        this.regno = regno;       }
    public String getDetails() {
        return "Name: " + this.name + ", Registration Number: " + this.regno;       }
    public void setDetails(String name, String regno) {
        this.name = name;
        this.regno = regno;       }
    public static void main(String[] args) {
        // Creating two Student objects
        Student student1 = new Student();
        Student student2 = new Student("Mark Reinhold", "MR002");
        // Setting details for both students
        student1.setDetails("James Gosling", "JG001");
        // Printing details of both students
        System.out.println("Student 1 Details: " + student1.getDetails());
        System.out.println("Student 2 Details: " + student2.getDetails());
    }   }
```

```
Student 1 Details: Name: James Gosling, Registration Number: JG001
Student 2 Details: Name: Mark Reinhold, Registration Number: MR002
```

```java
public class Student {
    private String name,regno; // Instance variables
    // Default constructor
    public Student() { // Initialize with default values
        name = "Unknown";
        regno = "0000";      }
    public Student(String name, String regno) { // Parameterized constructor
        name = name; //initialize without using this
        regno = regno;      }
    public String getDetails() {
      return "Name: " + this.name + ", Registration Number: " + this.regno;      }
    public void setDetails(String name, String regno) {
      this.name = name;
      this.regno = regno;      }
    public static void main(String[] args) {
      // Creating two Student objects
      Student student1 = new Student();
      Student student2 = new Student("Mark Reinhold", "MR002");
      // Setting details for both students
      student1.setDetails("James Gosling", "JG001");
      // Printing details of both students
      System.out.println("Student 1 Details: " + student1.getDetails());
      System.out.println("Student 2 Details: " + student2.getDetails());
    }   }
```

```
Student 1 Details: Name: James Gosling, Registration Number: JG001
Student 2 Details: Name: null, Registration Number: null
```

this

- ‘this’ is a keyword used as a special reference variable that refers to the current object of the class
- Used to invoke current class method

```
public class Counter {
   private int count; // Instance variable to store the counter value
   public Counter() {// Constructor
      this.count = 0; }
  public void increment() {// Method to increment the counter
      this.count++;
      this.displayCount(); } // Call displayCount method using 'this'
 public void decrement() {// Method to decrement the counter
      this.count--; // Decrement the count
      this.displayCount(); } // Call displayCount method using 'this'
 public void displayCount() {// Method to display the current count
      System.out.println("Current count: " + this.count);
   }
   public static void main(String[] args) {
     Counter counter = new Counter();
      counter.increment(); // Increment count to 1
      counter.increment(); // Increment count to 2
      counter.decrement(); // Decrement count to 1
      counter.decrement(); // Decrement count to 0
   }
}
```

```
Current count: 1
Current count: 2
Current count: 1
Current count: 0
```

```java
public class Counter{
   private int count; // Instance variable to store the counter value
   public Counter() {// Constructor
      this.count = 0; }
   public void increment() {// Method to increment the counter
      this.count++;
      displayCount(); } //  same as using this. displayCount()
   public void decrement() {// Method to decrement the counter
      this.count--; // Decrement the count
      displayCount(); } // same as using this. displayCount()
   public void displayCount() {// Method to display the current count
      System.out.println("Current count: " + this.count);
   }
   public static void main(String[] args) {
      Counter counter = new Counter();
      counter.increment(); // Increment count to 1
      counter.increment(); // Increment count to 2
      counter.decrement(); // Decrement count to 1
      counter.decrement(); // Decrement count to 0
   }
}
```

```
Current count: 1
Current count: 2
Current count: 1
Current count: 0
```

```
public class Car{
   private String model;

   public Car setModel(String model) {
      this.model = model;
      return this; // Returns the current object for method chaining
   }

   public void showModel() {
      System.out.println("Model: " + this.model); // Refers to the instance variable
   }

  public static void main(String[] args) {
         new Car().setModel("Tesla").showModel(); // Method chaining
      }
   }
```

```
Model: Tesla
```

this

- ‘this’ is a keyword used as a special reference variable that refers to the current object of the class
- Used to invoke current class constructor
    - Used to reuse the constructor
    - Constructor chaining

```
public class Car{
    private String model;
    public Car() {  // Default constructor
        this("Unknown"); // Calls the parameterized constructor with a default model
    }
    public Car(String model) {  // Parameterized constructor
        this.model = model;     // Sets the model
    }
    public Car setModel(String model) {
        this.model = model;
        return this;        // Returns the current object for method chaining
    }
    public void showModel() {
        System.out.println("Model: " + this.model); // Refers to the instance variable
    }
    public static void main(String[] args) {
        // Using the parameterized constructor
        Car car1 = new Car("Tesla");
        car1.showModel();       // Output: Model: Tesla
        // Using the default constructor and then setting the model
        Car car2 = new Car();     // Calls the default constructor
        //car2.setModel("BMW");    // Sets the model
        car2.showModel();        // Output: Model: BMW ?
    }   }
```

```
Model: Tesla
Model: Unknown
```

```
public class Student {
    private String name,regno; // Instance variables
    public Student() { // Default constructor
      name = "Unknown";
      regno = "0000";       }
    public Student(String name) { // Parameterized constructor
      this.name = name;
    }
    public Student(String name, String regno) { // Parameterized constructor
      this(name); //reusing constructor from the constructor
      this.regno = regno;
    }
    public String getDetails() {
      return "Name: " + this.name + ", Registration Number: " + this.regno;
    }
    public static void main(String[] args) {
      Student student1 = new Student();
      Student student2 = new Student("James Gosling");
      Student student3 = new Student("Mark Reinhold", "MR002");
      // Printing details of both students
      System.out.println("Student 1 Details: " + student1.getDetails());
      System.out.println("Student 2 Details: " + student2.getDetails());
      System.out.println("Student 2 Details: " + student3.getDetails());      }
  }
```

```
Student 1 Details: Name: Unknown, Registration Number: 0000
Student 2 Details: Name: James Gosling, Registration Number: null
Student 2 Details: Name: Mark Reinhold, Registration Number: MR002
```

```
public class Student {
      private String name,regno; // Instance variables
      public Student() { // Default constructor
         name = "Unknown";
         regno = "0000";        }
      public Student(String name) { // Parameterized constructor
         this.name = name;
      }
      public Student(String name, String regno) { // Parameterized constructor
        this.regno = regno;
        this(name); //reusing constructor from the constructor
      }
      public String getDetails() {
         return "Name: " + this.name + ", Registration Number: " + this.regno;
      }
      public static void main(String[] args) {
         Student student1 = new Student();
         Student student2 = new Student("James Gosling");
         Student student3 = new Student("Mark Reinhold", "MR002");
         // Printing details of both students
         System.out.println("Student 1 Details: " + student1.getDetails());
         System.out.println("Student 2 Details: " + student2.getDetails());
         System.out.println("Student 2 Details: " + student3.getDetails());      }
   }
```

```
Student 1 Details: Name: Unknown, Registration Number: 0000
Student 2 Details: Name: James Gosling, Registration Number: null
Student 2 Details: Name: Mark Reinhold, Registration Number: MR002
```

```
public class Student {
    private String name,regno; // Instance variables
    public Student() { // Default constructor
      name = "Unknown";
      regno = "0000";        }
    public Student(String name) { // Parameterized constructor
      this.name = name;
    }
    public Student(String name, String regno) { // error
     this.regno = regno;
     this(name); // this()should be the first statement
    }
    public String getDetails() {
      return "Name: " + this.name + ", Registration Number: " + this.regno;
    }
    public static void main(String[] args) {
      Student student1 = new Student();
      Student student2 = new Student("James Gosling");
      Student student3 = new Student("Mark Reinhold", "MR002");
      // Printing details of both students
      System.out.println("Student 1 Details: " + student1.getDetails());
      System.out.println("Student 2 Details: " + student2.getDetails());
      System.out.println("Student 2 Details: " + student3.getDetails());      }
  }
```

```
//compile time error
java: call to this must be first statement in constructor
```

```
public class Student {
    private String name,regno; // Instance variables
    public Student() { // Default constructor
        name = "Unknown";
        regno = "0000";        }
    public Student(String name) { // Parameterized constructor
        this.name = name;
    }
    public Student(String name, String regno) { // Parameterized constructor
        this();   //calling default constructor
    }
    public String getDetails() {
        return "Name: " + this.name + ", Registration Number: " + this.regno;
    }
    public static void main(String[] args) {
        Student student1 = new Student();
        Student student2 = new Student("James Gosling");
        Student student3 = new Student("Mark Reinhold", "MR002");
        // Printing details of both students
        System.out.println("Student 1 Details: " + student1.getDetails());
        System.out.println("Student 2 Details: " + student2.getDetails());
        System.out.println("Student 2 Details: " + student3.getDetails());       }
  }
```

```
Student 1 Details: Name: Unknown, Registration Number: 0000
Student 2 Details: Name: James Gosling, Registration Number: null
Student 2 Details: Name: Unknown, Registration Number: 0000
```

```
class Person {
  Student obj;

  Person(Student obj) {
    this.obj = obj;
  }

  public void printStudentDetails() {
    System.out.println(obj.name);
  }
}

  class Student {

  public String name="James"; // Instance variables
  public Student(){
    Person objP=new Person(this);
    objP.printStudentDetails();
  }

  public static void main(String[] args) {
   Student student = new Student();
     }
}
```

```
James
```

this

- ‘this’ is a keyword used as a special reference variable that refers to the current object of the class
- Used to return current class instance

```
class Student {

   public String name; // Instance variables
   public Student(String name){
      this.name=name;
   }
   public Student sendObject(){
      return this;
   }

   public String toString(){
         return "Name::" + name;
   }
   public static void main(String[] args) {
     Student obj=new Student("Jayachitra");
      System.out.println(obj.sendObject());
      }
}
```

```
Name::Jayachitra
```

```
  class Student {

   public String name; // Instance variables
   public Student(String name){
     this.name=name;
   }
   public void printObject(){
     System.out.println(this);
   }


   public static void main(String[] args) {
     Student obj=new Student("Jayachitra");
      System.out.println(obj);
      obj.printObject();
       }
}
```

```
Student@e9e54c2
Student@e9e54c2
```

```
class Sample {
   int x;
   Sample(int x) {
      this.x = x;            }
   void modify(Sample obj) {
      obj.x = 100;    }
   void reassign(Sample obj) {
      obj = new Sample(200);     }
}
public class Main {
   public static void main(String[] args) {
      Sample original = new Sample(10);
      System.out.println(" x before modify: " + original.x);
      original.modify(original);
      System.out.println("x after modify: " + original.x);
      original.reassign(original);
      System.out.println("x after reassign: " + original.x);
   }
}
```

```
class Sample {
   int x;
   Sample(int x) {
      this.x = x;
   }
   void modify(Sample obj) {
      obj.x = 100;    }
   void reassign(Sample obj) {
      obj = new Sample(200);       }
}
public class Main {
   public static void main(String[] args) {
      Sample original = new Sample(10);
      System.out.println(" x before modify: " + original.x);
      original.modify(original);
      System.out.println(" x after modify: " + original.x);
      original.reassign(original);
      System.out.println("x after reassign: " + original.x);
   }
}
```

```java
class Sample {
   int x;
   Sample(int x) {
      this.x = x;            }
   void modify(Sample obj) {
      obj.x = 100;    }
   void reassign(Sample obj) {
      obj = new Sample(200);     }
}
public class Main {
   public static void main(String[] args) {
      Sample original = new Sample(10);
      System.out.println(" x before modify: " + original.x);
      original.modify(original);
      System.out.println("x after modify: " + original.x);
      original.reassign(original);
      System.out.println("x after reassign: " + original.x);
   }
}
```

```
x before modify: 10
x after modify: 100
x after reassign: 100
```

```cpp
#include <iostream>
class Sample {
public:
   int x;
   Sample(int x) : x(x) {}
   void modify(Sample& obj) {
      obj.x = 100;    }
   void reassign(Sample*& obj) {
      obj = new Sample(200);    }};
int main() {
   Sample original(10);
   std::cout << " x before modify: " << original.x << std::endl;
   original.modify(original);
   std::cout << "x after modify: " << original.x << std::endl;
   Sample* originalPtr = &original;
   original.reassign(originalPtr);
   std::cout << "x after reassign: " << original.x << std::endl;
   delete originalPtr;
   return 0;
}
```

```
x before modify: 10
x after modify: 100
x after reassign: 200
```