l2-introduction-to-JAVA

July 22, 2024

1 Lecture notes on JAVA

By Jason Pandian *Assistant Professor, Department of Information Technology*

2 INTRODUCTION TO JAVA

Introduction to Java Programming –Features of Java- Classes and Objects - Arrays – Methods – Constructor - Access Specifier – Static members - Command Line Arguments - Strings Handling - Method Overloading– Method Overriding - Inheritance

2.1 Introduction to Java Programming

Java is a high-level, object-oriented programming language developed by Sun Microsystems in 1995. It is platform-independent, secure, and robust, making it widely used for building various types of applications.

2.1.1 Features of Java

1. **Simple**: Java is easy to learn and its syntax is clean and easy to understand.
2. **Object-Oriented**: Java is based on the object-oriented programming (OOP) paradigm, which promotes modular and reusable code.
3. **Platform-Independent**: Java programs can run on any device that has the Java Virtual Machine (JVM).
4. **Secure**: Java has built-in security features to protect against viruses and tampering.
5. **Robust**: Java has strong memory management and exception handling features.
6. **Multithreaded**: Java supports concurrent execution of multiple threads.
7. **Portable**: Java code is portable across different platforms without modification.
8. **High Performance**: Just-In-Time (JIT) compilers help improve the performance of Java applications.

2.1.2 Classes and Objects

Java is a purely object-oriented programming language. Everything in Java is associated with classes and objects.

“‘java public class Person { // Fields String name; int age;

```
// Method
void displayInfo() {
    System.out.println("Name: " + name);
    System.out.println("Age: " + age);
}

// Main method
public static void main(String[] args) {
    // Creating an object
    Person person = new Person();
    person.name = "Alice";
    person.age = 30;
    person.displayInfo();
}
```

}

2.2 Arrays

Arrays are used to store multiple values of the same type in a single variable.

```
public class ArrayExample {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};

        // Accessing array elements
        for (int i = 0; i < numbers.length; i++) {
            System.out.println(numbers[i]);
        }
    }
}
```

2.2.1 Methods

Methods are blocks of code that perform a specific task and are defined within a class.

```
public class Rectangle {
    int width, height;

    // Constructor
    Rectangle(int w, int h) {
        width = w;
        height = h;
    }
```

```
    int area() {
        return width * height;
    }

    public static void main(String[] args) {
        Rectangle rect = new Rectangle(10, 5);
        System.out.println("Area: " + rect.area());
    }
}
```

2.2.2 Constructor

A constructor is a special method that is called when an object is instantiated.

```
public class Rectangle {
    int width, height;

    // Constructor
    Rectangle(int w, int h) {
        width = w;
        height = h;
    }

    int area() {
        return width * height;
    }

    public static void main(String[] args) {
        Rectangle rect = new Rectangle(10, 5);
        System.out.println("Area: " + rect.area());
    }
}
```

2.2.3 Access Specifiers

Access specifiers define the accessibility of classes, methods, and variables. The main access specifiers in Java are public, protected, default (no keyword), and private.

```
public class AccessSpecifierExample {
    public int publicVar;
    protected int protectedVar;
    int defaultVar; // Default access
    private int privateVar;

    public void display() {
        System.out.println("Public: " + publicVar);
        System.out.println("Protected: " + protectedVar);
```

```
        System.out.println("Default: " + defaultVar);
        System.out.println("Private: " + privateVar);
    }
}
```

2.2.4 Static Members

Static members belong to the class rather than any specific instance. They are shared among all instances of the class.

```
public class StaticExample {
    static int staticVar = 10;

    static void display() {
        System.out.println("Static Variable: " + staticVar);
    }

    public static void main(String[] args) {
        StaticExample.display();
    }
}
```

2.2.5 Command Line Arguments

Java programs can accept command-line arguments.

```
public class CommandLineArgs {
    public static void main(String[] args) {
        for (String arg : args) {
            System.out.println(arg);
        }
    }
}
```

2.2.6 String Handling

Strings are objects in Java and are handled using the String class.

```
public class StringExample {
    public static void main(String[] args) {
        String str = "Hello, World!";
        System.out.println("Length: " + str.length());
        System.out.println("Substring: " + str.substring(0, 5));
        System.out.println("Uppercase: " + str.toUpperCase());
    }
}
```

2.2.7 Method Overloading

Method overloading allows multiple methods with the same name but different parameters.

```java
public class OverloadingExample {
    void display(int a) {
        System.out.println("Argument: " + a);
    }

    void display(String a) {
        System.out.println("Argument: " + a);
    }

    public static void main(String[] args) {
        OverloadingExample obj = new OverloadingExample();
        obj.display(10);
        obj.display("Hello");
    }
}
```

2.2.8 Method Overriding

Method overriding allows a subclass to provide a specific implementation of a method already defined in its superclass.

```java
class Animal {
    void sound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }

    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.sound();
    }
}
```

2.2.9 Inheritance

Inheritance allows one class to inherit the properties and methods of another class.

```java
class Person {
    String name;
    int age;

    void displayInfo() {
        System.out.println("Name: " + name);
```

```
        System.out.println("Age: " + age);
    }
}

class Student extends Person {
    String major;

    void displayMajor() {
        System.out.println("Major: " + major);
    }

    public static void main(String[] args) {
        Student student = new Student();
        student.name = "Bob";
        student.age = 20;
        student.major = "Computer Science";
        student.displayInfo();
        student.displayMajor();
    }
}
```

3 Any Questions or Doubts?

Refer the Lectures/Tutorials GitHub Page

[ ]: