JAVA PROGRAMMING BASICS

Module 2: Java Object-oriented Programming

Training program

1. Classes and Instances
2. The Methods
3. The Constructors
4. Static elements
5. Initialization sections
6. Package
7. Inheritance and Polymorphism
8. Abstract classes and interfaces
9. String processing
10. Exceptions and Assertions
11. Nested classes
12. Enums
13. Wrapper classes for primitive types
14. Generics
15. Collections
16. Method overload resolution
17. Multithreads
18. Core Java Classes
19. Object Oriented Design

Module contents

- The Constructors
  - Constructors declaration
  - Default Constructors
  - Constructors overloading
  - Calling one constructor from another

Module contents

- The Constructors
  - Constructors declaration
  - Default Constructors
  - Constructors overloading
  - Calling one constructor from another

Constructor declaration 1/4

```java
public class Car {
    private int maxSpeed;
    //...
    public Car(int maxSpeed) {
        this.maxSpeed = maxSpeed;
    }
    //...
    public int getMaxSpeed() {
        return maxSpeed;
    }
}
```

Constructor declaration

Constructor declaration 2/4

- Constructors are special methods defined in a class that create and return an object of the class in which they're defined.

Access modifier

Constructor parameters

```
public Car(int maxSpeed) {
        this.maxSpeed = maxSpeed;
  }
```

Constructor body

Constructor declaration 3/4

Get a block of unused memory in the heap, large enough to hold an object of the specified type

Call the default constructor of the superclass if no constructor is defined

Initialize member variables to the specified values (or default initial value is used)

Executes the body of the constructor

Constructor declaration 4/4

```java
public class Main {
    public static void main(String[] arg) {
        Car myCar1 = new Car(100);
        Car myCar2 = new Car(120);
        Car myCar3 = new Car(160);
        System.out.println(myCar1.getMaxSpeed());
        System.out.println(myCar2.getMaxSpeed());
        System.out.println(myCar3.getMaxSpeed());
    }
}
```

Module contents

- The Methods
  - Constructors declaration
  - Default Constructors
  - Constructors overloading
  - Calling one constructor from another

Default Constructors 1/4

- When we don’t define any constructor, the compiler creates the default constructor

```
public class Car {
    private int maxSpeed;
    //...
    public int getMaxSpeed() {
        return maxSpeed;
    }
    //...
}
```

- *equivalent to the declaration:* **public** Car() { }

Default Constructors 2/4

```java
public class Main {
    public static void main(String[] arg) {
        Car myCar1 = new Car();
        Car myCar2 = new Car();
        Car myCar3 = new Car();
        System.out.println(myCar1.getMaxSpeed());
        System.out.println(myCar2.getMaxSpeed());
        System.out.println(myCar3.getMaxSpeed());
    }
}
```

compiler create a no-arg default constructor

Default Constructors 3/4

- When we define any constructor, the compiler don’t creates the default constructor

```java
public class Car {
    private int maxSpeed;
    //...
    public Car(int maxSpeed) {
        this.maxSpeed = maxSpeed;
    }
    //...
    public int getMaxSpeed() {
        return maxSpeed;
    }
}
```

Default Constructors 4/4

```
public class Main {
    public static void main(String[] arg)
        Car myCar1 = new Car();
        Car myCar2 = new Car(100);
        Car myCar3 = new Car(120);
        System.out.println(myCar1.getMaxSpeed());
        System.out.println(myCar2.getMaxSpeed());
        System.out.println(myCar3.getMaxSpeed());
    }
}
```

compiler doesn't create
a no-arg constructor

Module contents

- The Methods
  - Constructors declaration
  - Default Constructors
  - Constructors overloading
  - Calling one constructor from another

Constructors overloading 1/4

```
public class Car {
    private int maxSpeed, milage;
    public Car() {
        this.maxSpeed = 100;
        this.milage = 1;
    }
    public Car(int maxSpeed) {
        this.maxSpeed = maxSpeed;
        this.milage = 1;
    }
    public Car(int maxSpeed, int milage) {
        this.maxSpeed = maxSpeed;
        this.milage = milage;
    }
}
```

Constructor with no arguments

Constructor with one argument

Constructor with two arguments

Constructors overloading 2/4

- The overloading is resolved at compile time by each class instance creation expression

```
public class Main {
    public static void main(String[] arg) {
        // Invokes no-argument constructor
        Car myCar1 = new Car();
        // Invokes one int argument constructor
        Car myCar2 = new Car(140);
        // Invokes two int argument constructor
        Car myCar3 = new Car(130, 10000);
    }
}
```

Constructors overloading 3/4

```java
public class Car {
    private int maxSpeed, milage;
    public Car() {
        this.maxSpeed = 100;
        this.milage = 1;
    }
    public Car(int maxSpeed) {
        this.maxSpeed = maxSpeed;
        this.milage = 1;
    }
    private Car(int maxSpeed, int milage) {
        this.maxSpeed = maxSpeed;
        this.milage = milage;
    }
}
```

Public Constructor with no arguments

Public Constructor with one argument

Private Constructor with two arguments

Constructors overloading 4/4

- The overloading is resolved at compile time by each class instance creation expression

```java
public class Main {
    public static void main(String[] arg) {
        // Invokes no-argument constructor
        Car myCar1 = new Car();
        // Invokes one int argument constructor
        Car myCar2 = new Car(140);
        // Invokes two int argument constructor
        Car myCar3 = new Car(130, 10000);
    }
}
```

private constructor cannot be called from anywhere outside the class

Module contents

- The Methods
  - Constructors declaration
  - Default Constructors
  - Constructors overloading
  - Calling one constructor from another

Calling one constructor from another 1/3

```java
public class Car {
    private int maxSpeed, milage;
    public Car() {
        this(100,1) ;
    }
    public Car(int maxSpeed) {
        this(maxSpeed,1);
    }
    public Car(int maxSpeed, int milage) {
        this.maxSpeed = maxSpeed;
        this.milage = milage;
    }
}
```

Calling one constructor from another 2/3

```java
public class Car {
    private int maxSpeed, milage;
    public Car() {
        System.out.println("Test Car()");
        this(100,1) ;
    }
    public Car(int maxSpeed) {
        this(maxSpeed,1);
        System.out.println("Test Car(int)");
    }
    public Car(int maxSpeed, int milage) {
        this.maxSpeed = maxSpeed;
        this.milage = milage;
    }
}
```

The first statement of a constructor body may be an explicit invocation of another constructor of the same class or of the direct superclass

Calling one constructor from another 3/3

```java
public class Car {
    private int maxSpeed, milage;
    public Car() {
        this(100,1) ;
    }
    public Car(int maxSpeed) {
        this(maxSpeed,1);
    }
    private Car(int maxSpeed, int milage) {
        this.maxSpeed = maxSpeed;
        this.milage = milage;
    }
}
```

A private constructor can still get called from other constructors