



CLASSES AND OBJECTS



Classes and Objects

Agenda

1

Classes & Objects

2

Static Block

Objectives

At the end of this module, you will be able to:

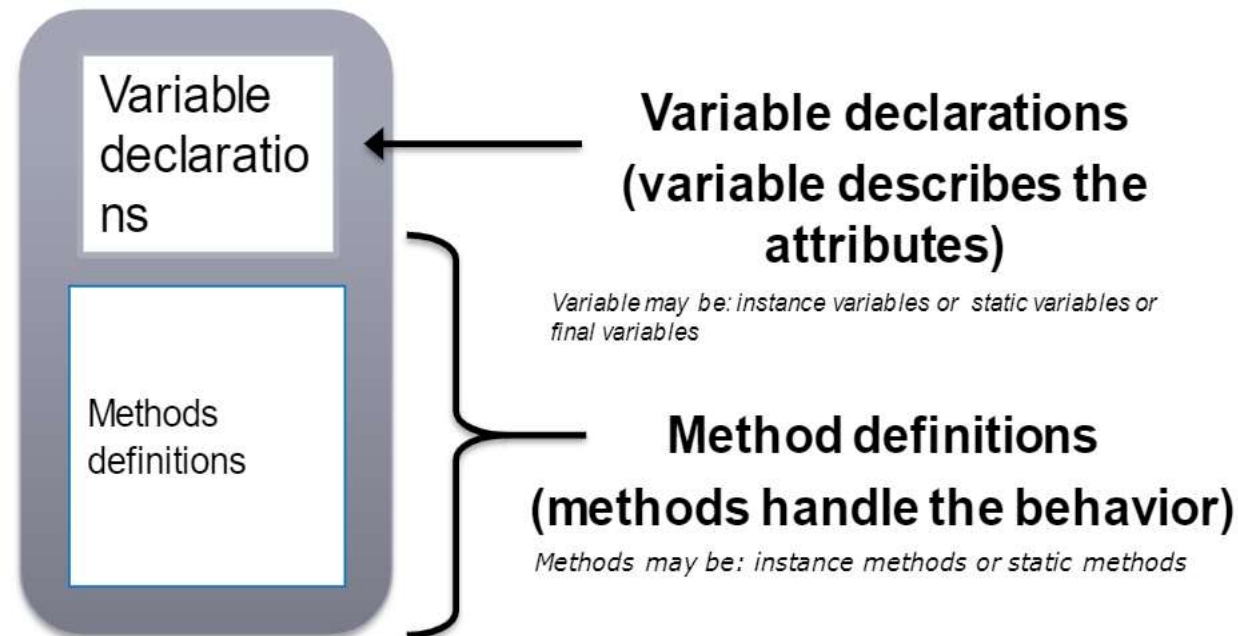
- Create classes and Objects
- Understand the importance of static block

Classes & Objects



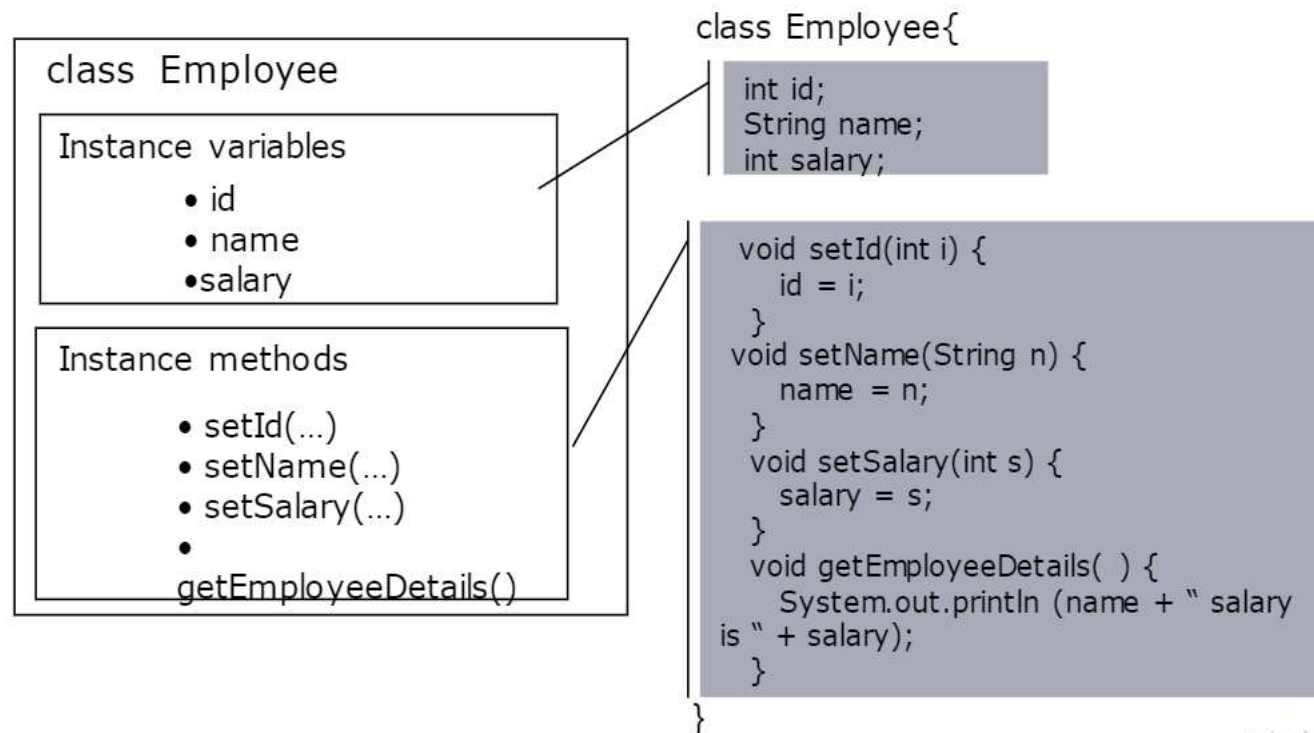
Classes

A class contains variable declarations and method definitions



Defining a Class in java

Define an Employee class with instance variables and instance methods



Basic information about a class

```
public class Account {  
    double balance;  
    public void deposit( double amount ){  
        balance += amount;  
    }  
    public double withdraw( double amount ){  
        int minimum_balance=5000;  
        if (balance >= (amount+minimum_balance)){  
            balance -= amount;  
            return amount;  
        }  
        else {  
            System.out.println("Insufficient Balance");  
            return 0.0;  
        }  
    }  
    public double getbalance(){  
        return balance;  
    }  
}
```

Instance
Variable

Parameter
or argument

local
Variable

Member variables

- The previous slide contains definition of a class called Accounts.
- A class contains members which can either be variables(fields) or methods(behaviors).
- A variable declared within a class(outside any method) is known as an **instance variable**.
- A variable declared within a method is known as **local variable**.
- Variables with method declarations are known as **parameters or arguments**.
- A class variable can also be declared as static where as a local variable cannot be static.

Objects and References

- Once a class is defined, you can declare a variable (object reference) of type class

Student stud1;

Employee emp1;

- The `new` operator is used to create an object of that reference type

Employee emp = new Employee();

↑
Object reference

↙
object

- Object references are used to store objects.
- Reference can be created for any type of classes (like concrete classes, abstract classes) and interfaces.

Objects and References (Contd.).

- The new operator,
 - Dynamically allocates memory for an object
 - Creates the object on the heap
 - Returns a reference to it
 - The reference is then stored in the variable

Employee class - Example

```
class Employee{
    int id;
    String name;
    int salary;
    void setId(int no){
        id = no;
    }
    void setName(String n){
        name = n;
    }
    void setSalary(int s){
        salary = s;
    }
    void getEmployeeDetails(){
        System.out.println(name + " salary is " + salary);
    }
}

public class EmployeeDemo {
    public static void main(String[] args) {
        Employee emp1 = new Employee();
        emp1.setId(101);
        emp1.setName("John");
        emp1.setSalary(12000);
        emp1.getEmployeeDetails();
    }
}
```

Output:

John salary is 12000

Constructors

- While designing a class, the class designer can define within the class, a special method called ‘constructor’
- Constructor is automatically invoked whenever an object of the class is created
- Rules to define a constructor
 - A constructor has the same name as the class name
 - A constructor should not have a return type
 - A constructor can be defined with any access specifier (like private, public)
 - A class can contain more than one constructor, So it can be overloaded

Constructor - Example

```
class Sample{  
    private int id;  
    Sample(){  
        id = 101;  
        System.out.println("Default constructor, with ID: "+id);  
    }  
    Sample(int no){  
        id = no;  
        System.out.println("One argument constructor,with ID: "+ id);  
    }  
}  
  
public class ConstDemo {  
    public static void main(String[] args) {  
        Sample s1 = new Sample();  
        Sample s2 = new Sample(102);  
    }  
}
```

Output:

Default constructor, with ID: 101
One argument constructor,with ID: 102

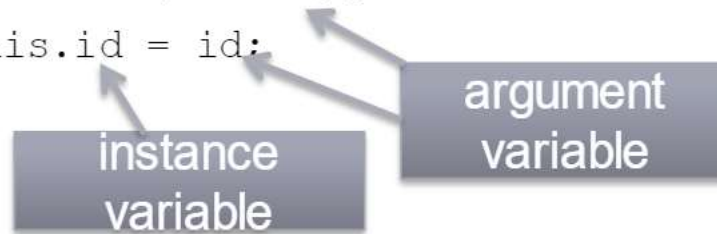
this reference keyword

- Each class member function contains an implicit reference of its class type, named this
- this reference is created automatically by the compiler
- It contains the address of the object through which the function is invoked
- Use of this keyword
 - this can be used to refer instance variables when there is a clash with local variables or method arguments
 - this can be used to call overloaded constructors from another constructor of the same class

this Reference (Contd.).

- Ex1:

```
void setId (int id){  
    this.id = id;  
}
```



- Ex2:

```
class Sample{  
    Sample(){  
        this("Java"); // calls overloaded constructor  
        System.out.println("Default constructor ");  
    }  
    Sample(String str){  
        System.out.println("One argument constructor "+ str);  
    }  
}
```


this Reference (Contd.).

- Use `this.variableName` to explicitly refer to the instance variable.
- Use `variable Name` to refer to the parameter.
- The **this** reference is implicitly used to refer to instance variables and methods.
- It **CANNOT** be used in a static method.

Static Class Members

- Static class members are the members of a class that do not belong to an instance of a class
- We can access static members directly by prefixing the members with the class name

ClassName.staticVariable

ClassName.staticMethod(...)

Static variables:

- Shared among all objects of the class
- Only one copy exists for the entire class to use

Static Class Members (Contd.).

- Stored within the class code, separately from instance variables that describe an individual object
- Public static final variables are global constants

Static methods:

- Static methods can only access directly the static members and manipulate a class's static variables
- Static methods cannot access non-static members(instance variables or instance methods) of the class
- Static method cant access this and super references

Static Class Members – Example

```
class StaticDemo
{
private static int a = 0;
private int b;
public void set ( int i, int j)
{
a = i; b = j;
}
public void show( )
{
System.out.println("This is static a: " + a );
System.out.println( "This is non-static b: " + b );
}
```

Static Class Members – Example (Contd.).

```
public static void main(String args[ ])  
{  
    StaticDemo x = new StaticDemo( );  
    StaticDemo y = new StaticDemo( );  
    x.set(1, 1);  
    x.show( );  
    y.set(2, 2);  
    y.show( );  
    x.show( );  
}
```

Output:

**This is static a: 1
This is non-static b: 1**

**This is static a: 2
This is non-static b: 2**

**This is static a: 2
This is non-static b: 1**

Quiz

- What will be the result, if we try to compile and execute the following code as

```
java Sample
class Sample{
    int i_val;
    public static void main(String[] xyz){
        System.out.println("i_val is :"+this.i_val);
    }
}
```

Quiz (Contd.).

- What will be the result, if we try to compile and execute the following code as

```
java Sample
class Sample{
    int i_val=10;
    Sample(int i_val){
        this.i_val=i_val;
        System.out.println("inside Sample i_val:
"+this.i_val);
    }
    public static void main(String[] xyz){
        Sample o = new Sample();
    }
```

Quiz- Solutions

- **Answer 1:**

Error: this keyword cannot be referred inside static methods.

- **Answer 2:**

Error: No default constructor created.

Static Block



The “static” block

- A static block is a block of code enclosed in braces, preceded by the keyword static

Ex :

```
static {  
    System.out.println("Within static block");  
}
```

- The statements within the static block are executed automatically when the class is loaded into JVM

The “static” block (Contd.).

- A class can have any number of static blocks and they can appear anywhere in the class
- They are executed in the order of their appearance in the class
- JVM combines all the static blocks in a class as single static block and executes them
- You can invoke static methods from the static block and they will be executed as and when the static block gets executed

Example on the “static” block (Contd.).

```
class StaticBlockExample {  
    StaticBlockExample() {  
        System.out.println("Within constructor");  
    }  
    static {  
        System.out.println("Within 1st static block");  
    }  
    static void m1() {  
        System.out.println("Within static m1 method");  
    }  
    static {  
        System.out.println("Within 2nd static block");  
        m1();  
    }  
}
```

Example on the “static” block (Contd.).

```
public static void main(String [] args) {  
    System.out.println("Within main");  
    StaticBlockExample x = new StaticBlockExample();  
}  
static {  
    System.out.println("Within 3rd static block");  
}  
}
```

Output:

Within 1st static block
Within 2nd static block
Within static m1 method
Within 3rd static block
Within main
Within constructor

Quiz

- What will be the result, if we try to compile and execute the following code as

java Sample

```
class Sample{  
    public static void main(String[] xyz){  
        System.out.println("Inside main method line1");  
    }  
    static {  
        System.out.println("Inside class line1");  
    }  
}
```

Quiz- Solution

- **Answer:**

Inside class line1

Inside main method line1

Summary

In this module, we were able to:

- Create classes and Objects
- Understand the importance of static block



Thank You