	BANGLADESH UNIVERSITY Department of Computer Science & Engineering **CSE2102 – Programming Language (Java) Lab**

PROJECT

SUBMITTED BY:

Student's Name	: Sheoti
Student ID	: 202221063026
Course Code	: CSE2102
Course Title	: Programming Language (Java) Lab

SUBMITTED TO:

Pabon Shaha

Lecturer

Department of Computer Science & Engineering

Bangladesh University

Date of Submission : 17/08/2023

Marks:

Real World Example: Account

➢ Object in JAVA:

- An entity that has a state and behavior is known as an object.
- e.g., chair, bike, marker, pen, table, car, etc.
- It can be physical or logical (tangible and intangible).
- The example of an intangible object is the banking system.

➢ Class in JAVA:

- A class is a group of objects which have common properties.
- It is a template or blueprint from which objects are created.
- It is a logical entity.
- It can't be physical.

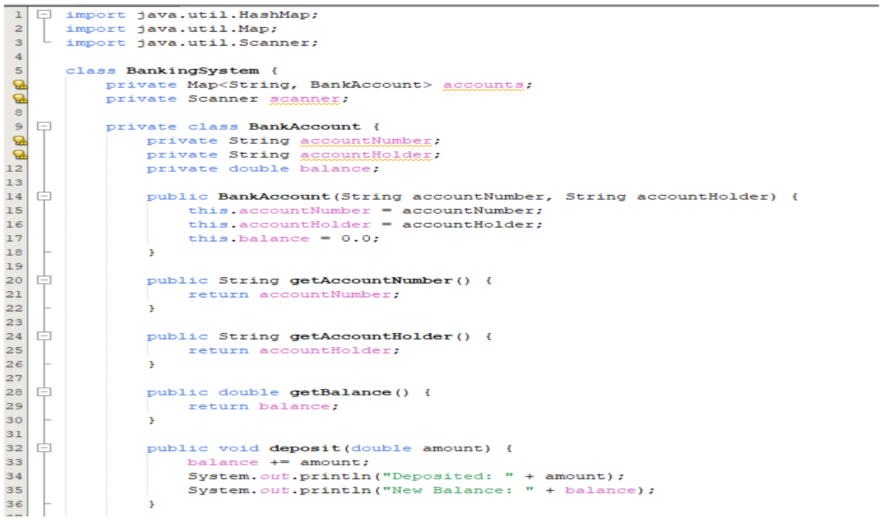
➢ Method in JAVA:

In Java, a method is like a function that is used to expose the behavior of an object.

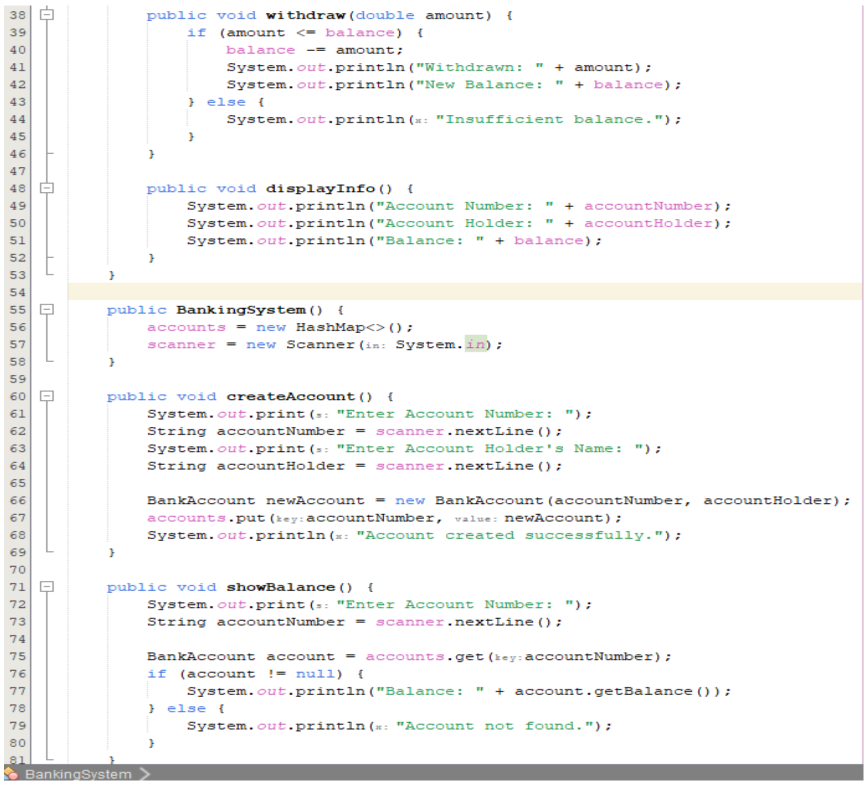
➢ Real Example:

First create Java Program to demonstrate the working of a banking system, where we deposit andwithdraw amounts from our account. Creating an Account class with deposit () and withdraw ()methods.

➢ Input:

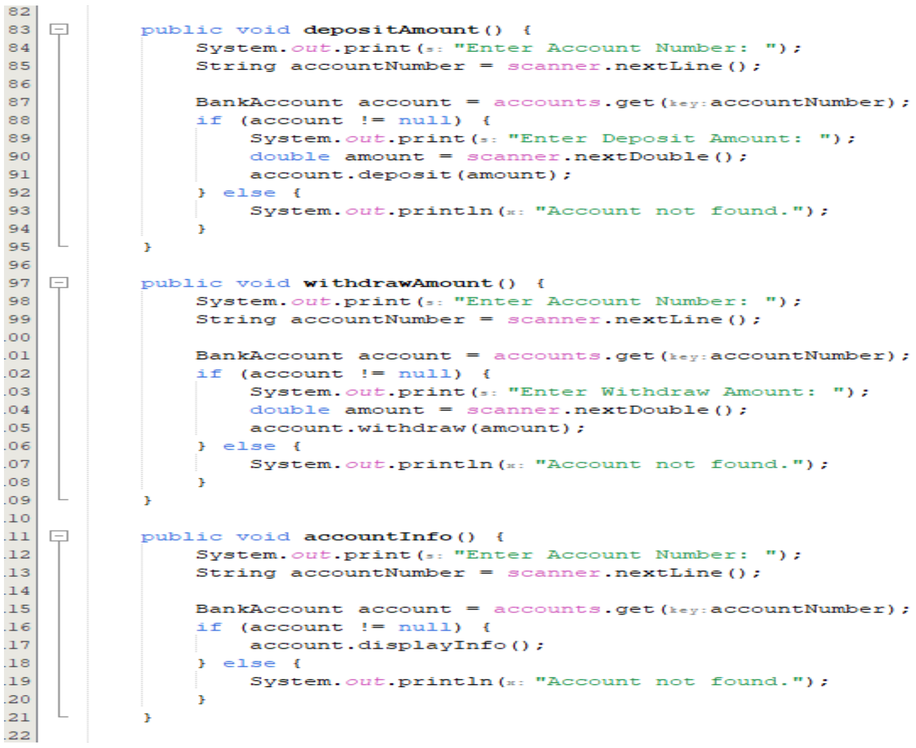


```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

class BankingSystem {
    private Map<String, BankAccount> accounts;
    private Scanner scanner;

    private class BankAccount {
        private String accountNumber;
        private String accountHolder;
        private double balance;

        public BankAccount(String accountNumber, String accountHolder) {
            this.accountNumber = accountNumber;
            this.accountHolder = accountHolder;
            this.balance = 0.0;
        }

        public String getAccountNumber() {
            return accountNumber;
        }

        public String getAccountHolder() {
            return accountHolder;
        }

        public double getBalance() {
            return balance;
        }

        public void deposit(double amount) {
            balance += amount;
            System.out.println("Deposited: " + amount);
            System.out.println("New Balance: " + balance);
        }
```

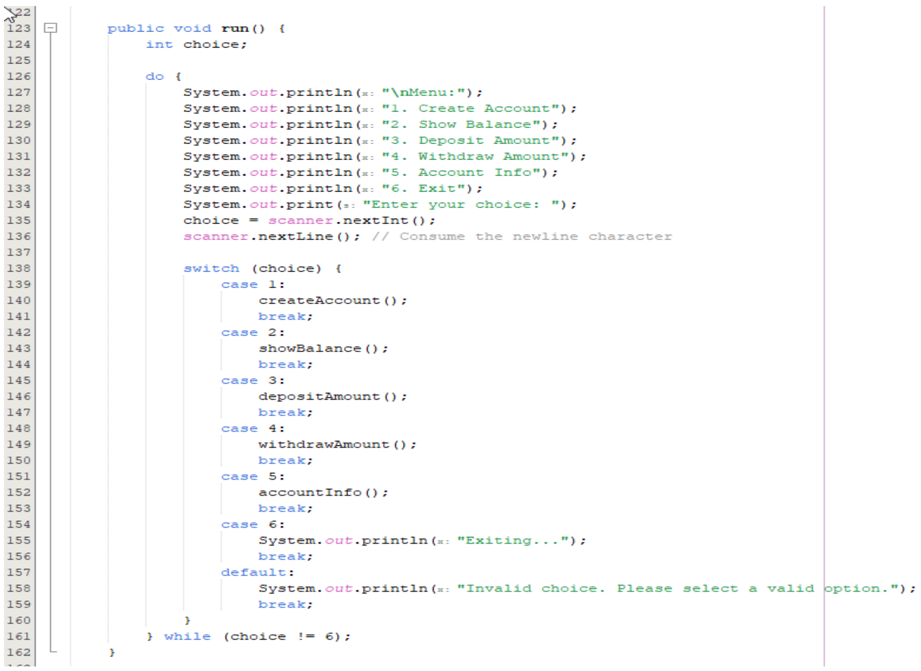


```java
        public void withdraw(double amount) {
            if (amount <= balance) {
                balance -= amount;
                System.out.println("Withdrawn: " + amount);
                System.out.println("New Balance: " + balance);
            } else {
                System.out.println(x: "Insufficient balance.");
            }
        }

        public void displayInfo() {
            System.out.println("Account Number: " + accountNumber);
            System.out.println("Account Holder: " + accountHolder);
            System.out.println("Balance: " + balance);
        }
    }

    public BankingSystem() {
        accounts = new HashMap<>();
        scanner = new Scanner(in: System.in);
    }

    public void createAccount() {
        System.out.print(s: "Enter Account Number: ");
        String accountNumber = scanner.nextLine();
        System.out.print(s: "Enter Account Holder's Name: ");
        String accountHolder = scanner.nextLine();

        BankAccount newAccount = new BankAccount(accountNumber, accountHolder);
        accounts.put(key:accountNumber, value: newAccount);
        System.out.println(x: "Account created successfully.");
    }

    public void showBalance() {
        System.out.print(s: "Enter Account Number: ");
        String accountNumber = scanner.nextLine();

        BankAccount account = accounts.get(key:accountNumber);
        if (account != null) {
            System.out.println("Balance: " + account.getBalance());
        } else {
            System.out.println(x: "Account not found.");
        }
    }
```

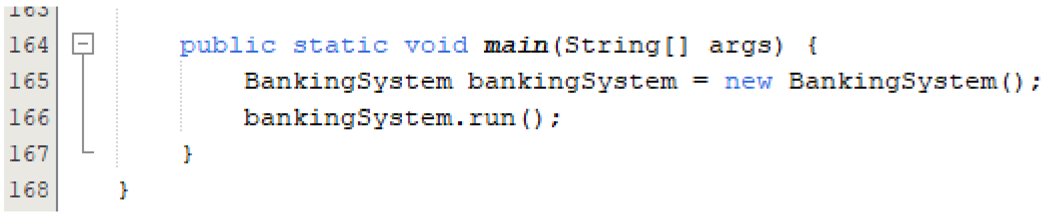
BankingSystem >



```java

    public void depositAmount() {
        System.out.print(s: "Enter Account Number: ");
        String accountNumber = scanner.nextLine();

        BankAccount account = accounts.get(key:accountNumber);
        if (account != null) {
            System.out.print(s: "Enter Deposit Amount: ");
            double amount = scanner.nextDouble();
            account.deposit(amount);
        } else {
            System.out.println(x: "Account not found.");
        }
    }

    public void withdrawAmount() {
        System.out.print(s: "Enter Account Number: ");
        String accountNumber = scanner.nextLine();

        BankAccount account = accounts.get(key:accountNumber);
        if (account != null) {
            System.out.print(s: "Enter Withdraw Amount: ");
            double amount = scanner.nextDouble();
            account.withdraw(amount);
        } else {
            System.out.println(x: "Account not found.");
        }
    }

    public void accountInfo() {
        System.out.print(s: "Enter Account Number: ");
        String accountNumber = scanner.nextLine();

        BankAccount account = accounts.get(key:accountNumber);
        if (account != null) {
            account.displayInfo();
        } else {
            System.out.println(x: "Account not found.");
        }
    }

```

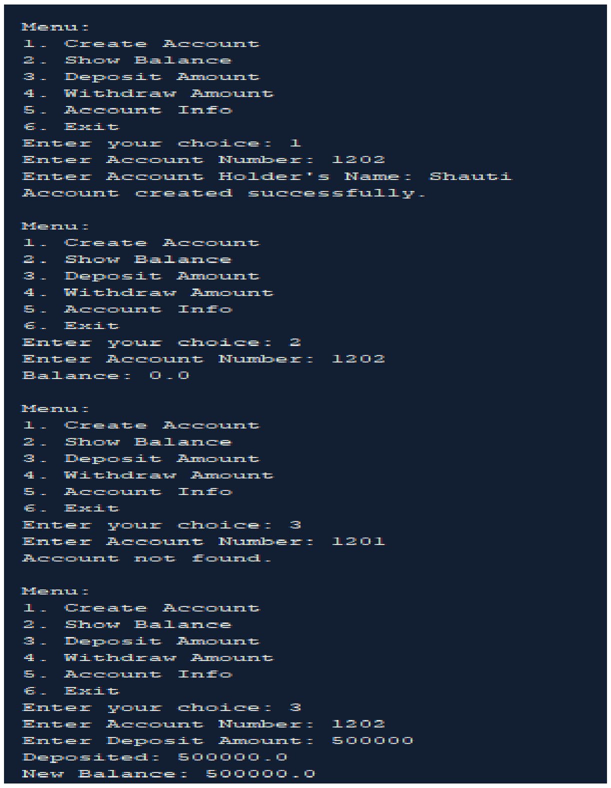


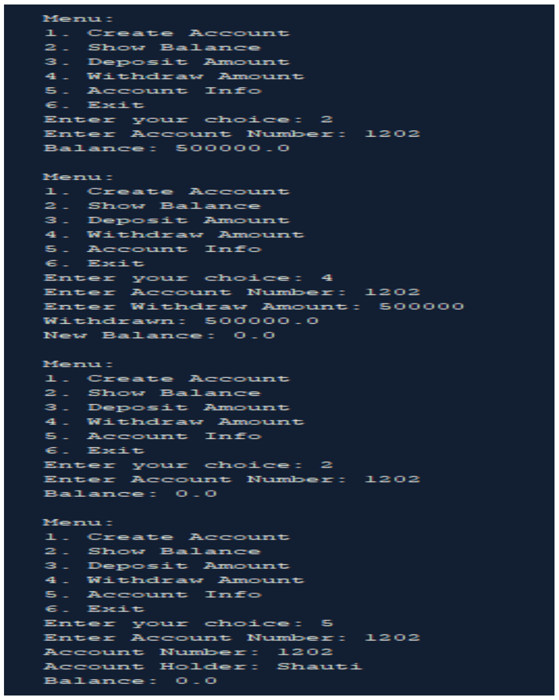
```java
    public void run() {
        int choice;

        do {
            System.out.println(x: "\nMenu:");
            System.out.println(x: "1. Create Account");
            System.out.println(x: "2. Show Balance");
            System.out.println(x: "3. Deposit Amount");
            System.out.println(x: "4. Withdraw Amount");
            System.out.println(x: "5. Account Info");
            System.out.println(x: "6. Exit");
            System.out.print(s: "Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline character

            switch (choice) {
                case 1:
                    createAccount();
                    break;
                case 2:
                    showBalance();
                    break;
                case 3:
                    depositAmount();
                    break;
                case 4:
                    withdrawAmount();
                    break;
                case 5:
                    accountInfo();
                    break;
                case 6:
                    System.out.println(x: "Exiting...");
                    break;
                default:
                    System.out.println(x: "Invalid choice. Please select a valid option.");
                    break;
            }
        } while (choice != 6);
    }
```

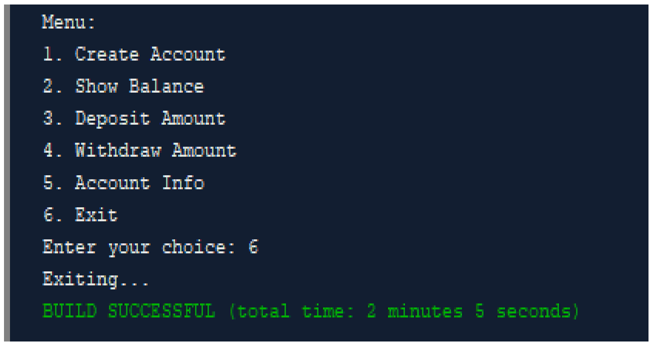


```java
    public static void main(String[] args) {
        BankingSystem bankingSystem = new BankingSystem();
        bankingSystem.run();
    }
}
```

➢ Output:


```
Menu:
1. Create Account
2. Show Balance
3. Deposit Amount
4. Withdraw Amount
5. Account Info
6. Exit
Enter your choice: 1
Enter Account Number: 1202
Enter Account Holder's Name: Shauti
Account created successfully.

Menu:
1. Create Account
2. Show Balance
3. Deposit Amount
4. Withdraw Amount
5. Account Info
6. Exit
Enter your choice: 2
Enter Account Number: 1202
Balance: 0.0

Menu:
1. Create Account
2. Show Balance
3. Deposit Amount
4. Withdraw Amount
5. Account Info
6. Exit
Enter your choice: 3
Enter Account Number: 1201
Account not found.

Menu:
1. Create Account
2. Show Balance
3. Deposit Amount
4. Withdraw Amount
5. Account Info
6. Exit
Enter your choice: 3
Enter Account Number: 1202
Enter Deposit Amount: 500000
Deposited: 500000.0
New Balance: 500000.0
```


```
Menu:
1. Create Account
2. Show Balance
3. Deposit Amount
4. Withdraw Amount
5. Account Info
6. Exit
Enter your choice: 2
Enter Account Number: 1202
Balance: 500000.0

Menu:
1. Create Account
2. Show Balance
3. Deposit Amount
4. Withdraw Amount
5. Account Info
6. Exit
Enter your choice: 4
Enter Account Number: 1202
Enter Withdraw Amount: 500000
Withdrawn: 500000.0
New Balance: 0.0

Menu:
1. Create Account
2. Show Balance
3. Deposit Amount
4. Withdraw Amount
5. Account Info
6. Exit
Enter your choice: 2
Enter Account Number: 1202
Balance: 0.0

Menu:
1. Create Account
2. Show Balance
3. Deposit Amount
4. Withdraw Amount
5. Account Info
6. Exit
Enter your choice: 5
Enter Account Number: 1202
Account Number: 1202
Account Holder: Shauti
Balance: 0.0
```


```
Menu:
1. Create Account
2. Show Balance
3. Deposit Amount
4. Withdraw Amount
5. Account Info
6. Exit
Enter your choice: 6
Exiting...
BUILD SUCCESSFUL (total time: 2 minutes 5 seconds)
```

Build a calculator by using class and object in Java

➢ Method in JAVA:

• A method is a block of code or collection of statements or a set of

codes grouped together to perform a certain task or operation.

• It is used to achieve the reusability of code.

• It also provides easy modification.

• And readability of code, just by adding or removing a chunk of code.

• The most important method in Java is the main () method.

➢ Naming a Method:

• Method name must be a verb and start with a lowercase letter.

• In the multi-word method name, the first letter of each word must be in uppercase except the first word.

• For example:

• Single-word method name: sum(), area()

• Multi-word method name: areaOfCircle (), stringComparision ()

➢ Types of Method:

There are two types of methods in Java:

I. Predefined Method

II. User-defined Method

➢ Example:

Here I build a calculator by using class and object in Java. Where methods are-

I. Addition

II. Subtraction

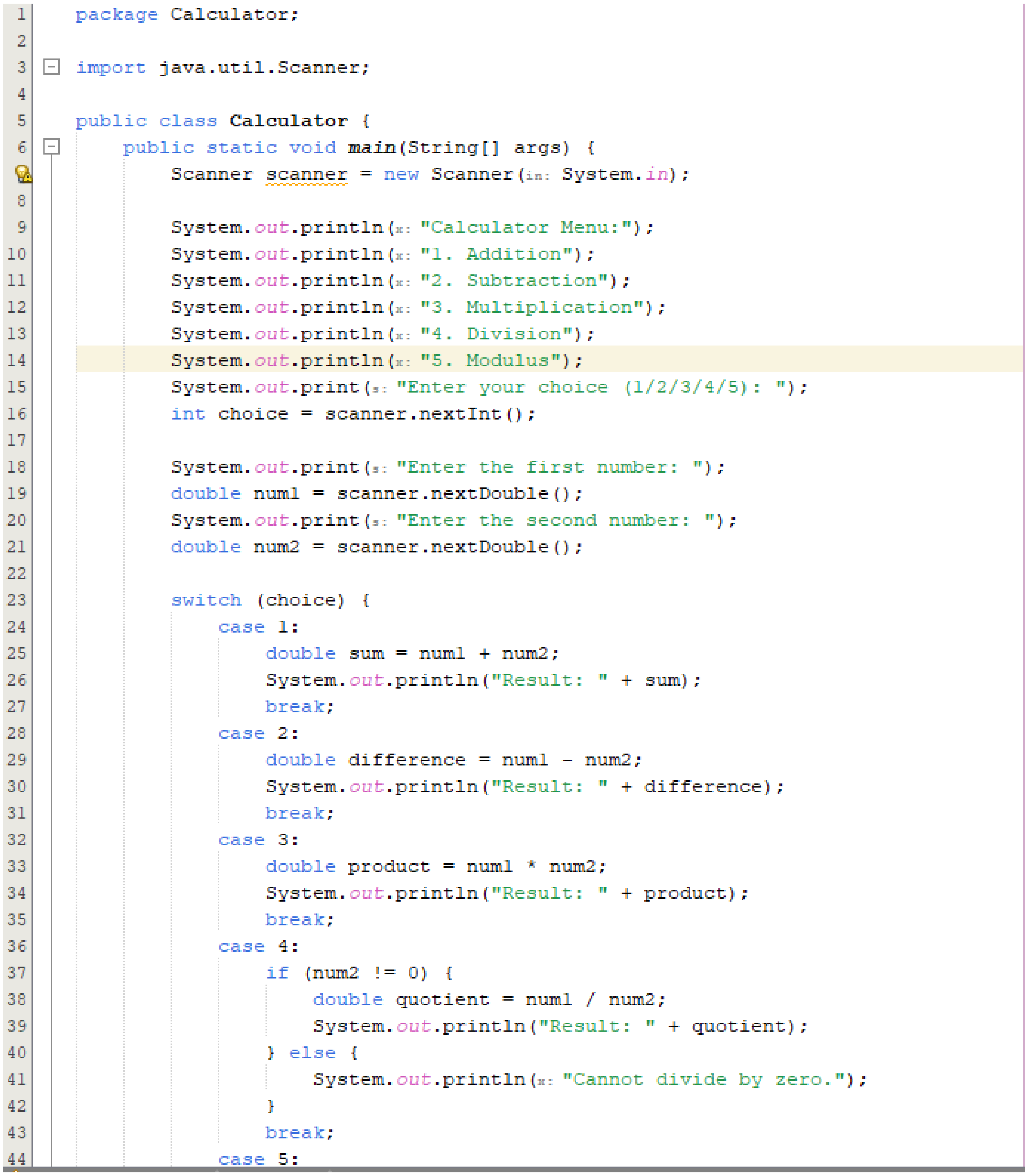
III. Multiplication

IV. Division

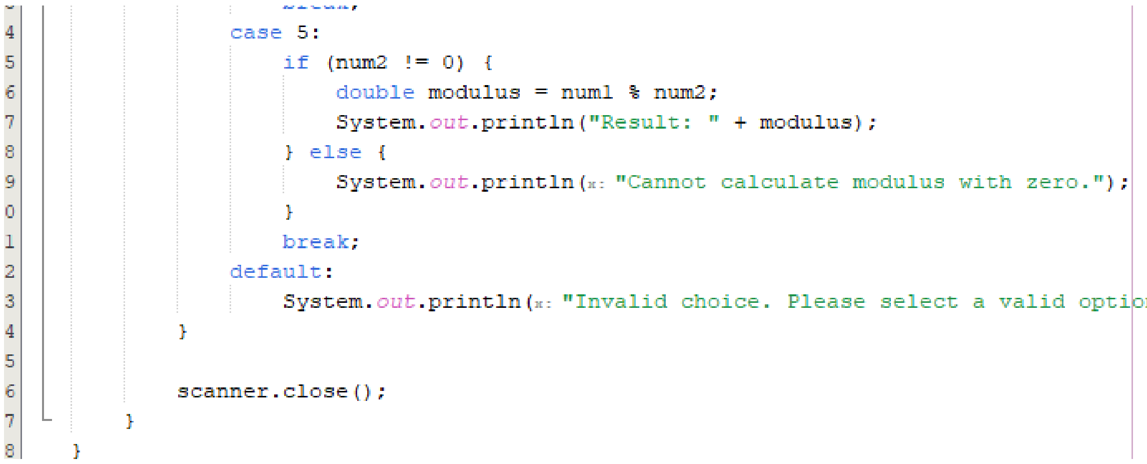
V. Mod

- Which take user choice and take user input from user.

➢ Code:

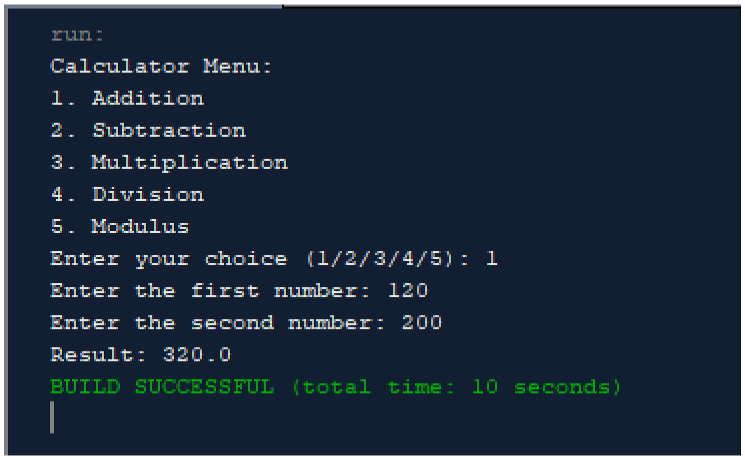


```
package Calculator;

import java.util.Scanner;

public class Calculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(in: System.in);

        System.out.println(x: "Calculator Menu:");
        System.out.println(x: "1. Addition");
        System.out.println(x: "2. Subtraction");
        System.out.println(x: "3. Multiplication");
        System.out.println(x: "4. Division");
        System.out.println(x: "5. Modulus");
        System.out.print(s: "Enter your choice (1/2/3/4/5): ");
        int choice = scanner.nextInt();

        System.out.print(s: "Enter the first number: ");
        double num1 = scanner.nextDouble();
        System.out.print(s: "Enter the second number: ");
        double num2 = scanner.nextDouble();

        switch (choice) {
            case 1:
                double sum = num1 + num2;
                System.out.println("Result: " + sum);
                break;
            case 2:
                double difference = num1 - num2;
                System.out.println("Result: " + difference);
                break;
            case 3:
                double product = num1 * num2;
                System.out.println("Result: " + product);
                break;
            case 4:
                if (num2 != 0) {
                    double quotient = num1 / num2;
                    System.out.println("Result: " + quotient);
                } else {
                    System.out.println(x: "Cannot divide by zero.");
                }
                break;
            case 5:
```



```java
            case 5:
                if (num2 != 0) {
                    double modulus = num1 % num2;
                    System.out.println("Result: " + modulus);
                } else {
                    System.out.println(x: "Cannot calculate modulus with zero.");
                }
                break;
            default:
                System.out.println(x: "Invalid choice. Please select a valid option.");
        }

        scanner.close();
    }
}
```

➢ **Output:**



```
run:
Calculator Menu:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Modulus
Enter your choice (1/2/3/4/5): 1
Enter the first number: 120
Enter the second number: 200
Result: 320.0
BUILD SUCCESSFUL (total time: 10 seconds)
```

Inheritance in JAVA

➢ Introduction:

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

➢ Types of Inheritance:

There are five types of inheritance.

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
4. Multiple Inheritance
5. Hybrid Inheritance

➢ Terms used in Inheritance:

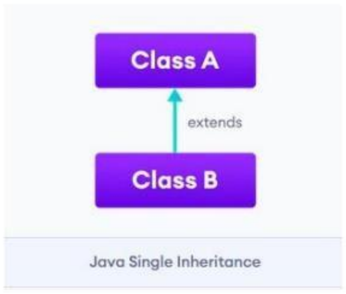
• **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.

• **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.

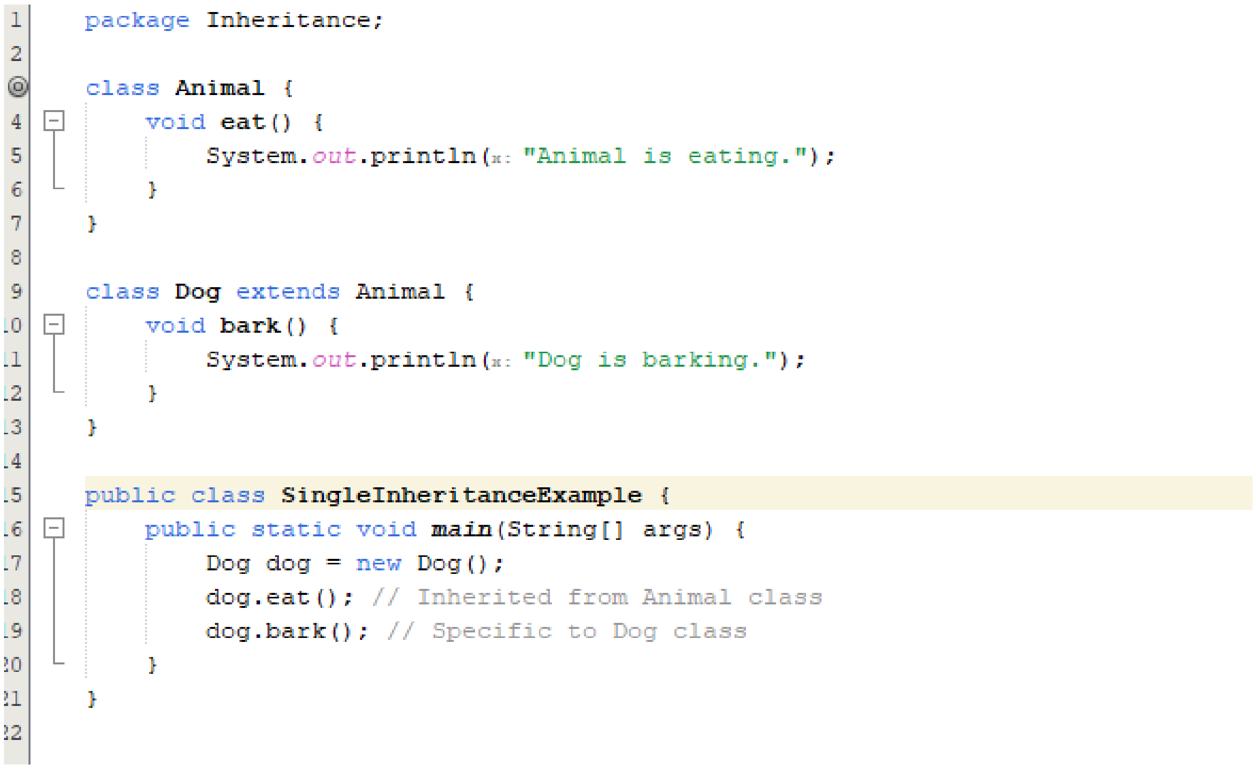
• **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

Single Inheritance: In single inheritance, a single subclass extends from a single superclass.

• **For example:**

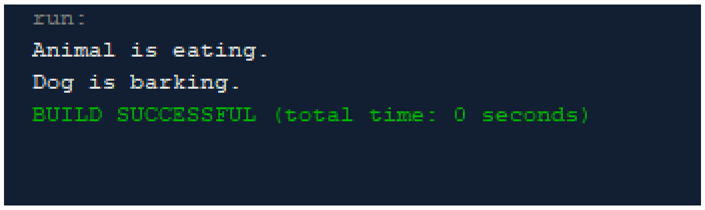


➢ Implementation of Single Inheritance:



```java
package Inheritance;

class Animal {
    void eat() {
        System.out.println(x: "Animal is eating.");
    }
}

class Dog extends Animal {
    void bark() {
        System.out.println(x: "Dog is barking.");
    }
}

public class SingleInheritanceExample {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat(); // Inherited from Animal class
        dog.bark(); // Specific to Dog class
    }
}
```

➢ Output:

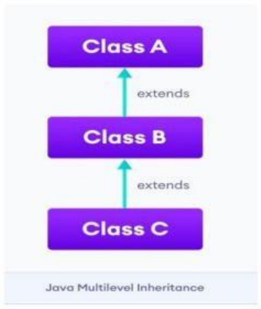


```
run:
Animal is eating.
Dog is barking.
BUILD SUCCESSFUL (total time: 0 seconds)
```

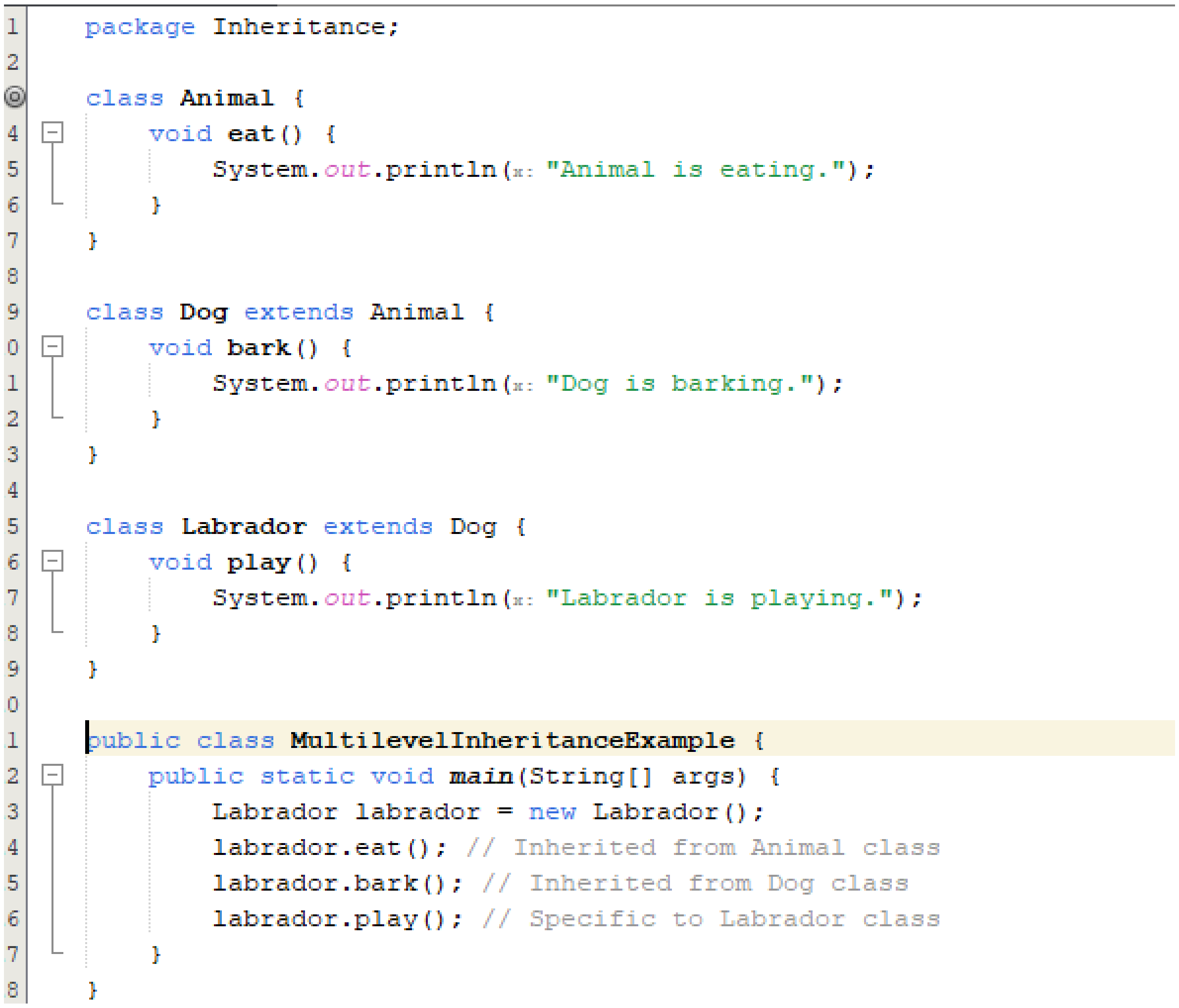
➢ Multilevel Inheritance:

In multilevel inheritance, a subclass extends from a superclass andthen the same subclass acts as a superclass for another class.

- **For example:**

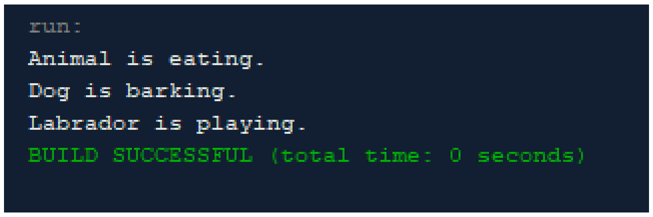


- **Implementation of Multilevel Inheritance:**



```java
package Inheritance;

class Animal {
    void eat() {
        System.out.println(x: "Animal is eating.");
    }
}

class Dog extends Animal {
    void bark() {
        System.out.println(x: "Dog is barking.");
    }
}

class Labrador extends Dog {
    void play() {
        System.out.println(x: "Labrador is playing.");
    }
}

public class MultilevelInheritanceExample {
    public static void main(String[] args) {
        Labrador labrador = new Labrador();
        labrador.eat(); // Inherited from Animal class
        labrador.bark(); // Inherited from Dog class
        labrador.play(); // Specific to Labrador class
    }
}
```

- **Output:**

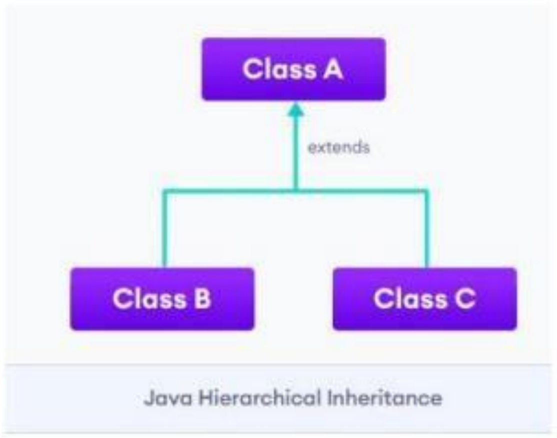


```
run:
Animal is eating.
Dog is barking.
Labrador is playing.
BUILD SUCCESSFUL (total time: 0 seconds)
```

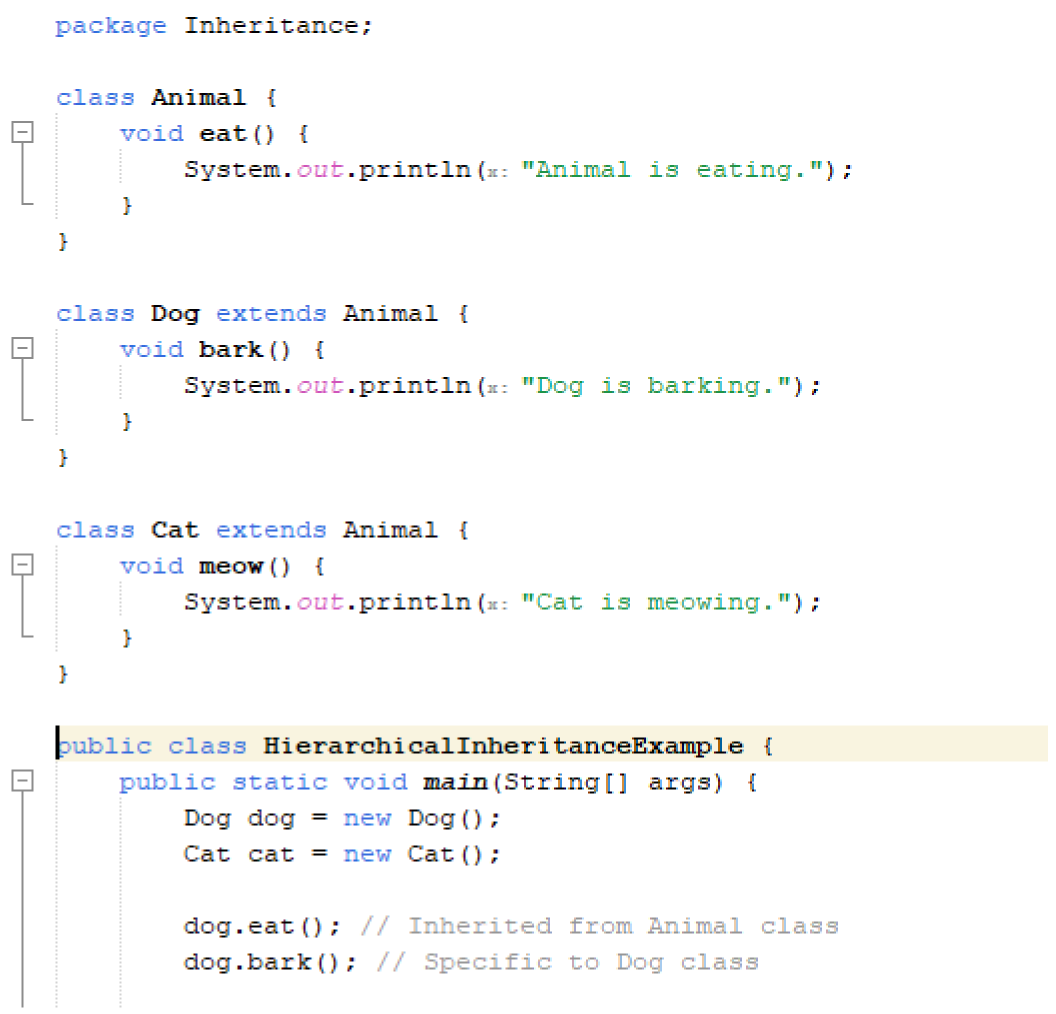
➢ Hierarchical Inheritance:

In hierarchical inheritance, multiple subclasses extend from asingle superclass.

- **For example:**

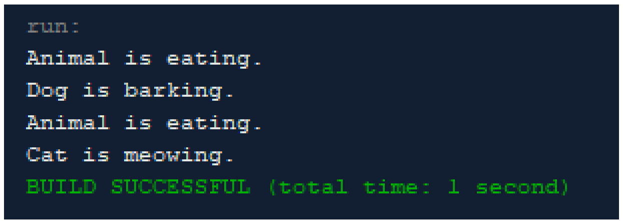


- **Implementation of Hierarchical Inheritance:**



```java
package Inheritance;

class Animal {
    void eat() {
        System.out.println(x: "Animal is eating.");
    }
}

class Dog extends Animal {
    void bark() {
        System.out.println(x: "Dog is barking.");
    }
}

class Cat extends Animal {
    void meow() {
        System.out.println(x: "Cat is meowing.");
    }
}

public class HierarchicalInheritanceExample {
    public static void main(String[] args) {
        Dog dog = new Dog();
        Cat cat = new Cat();

        dog.eat(); // Inherited from Animal class
        dog.bark(); // Specific to Dog class
        cat.eat(); // Inherited from Animal class
        cat.meow(); // Specific to Cat class
    }
}
```

- **Output:**

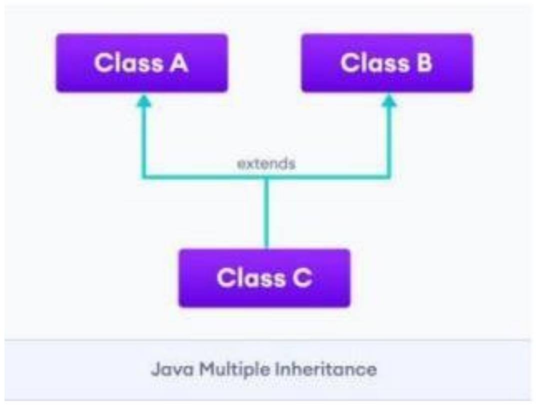


```
run:
Animal is eating.
Dog is barking.
Animal is eating.
Cat is meowing.
BUILD SUCCESSFUL (total time: 1 second)
```

➢ Multiple Inheritance:

In multiple inheritance, a single subclass extends from multiple super class.

- **For example:**

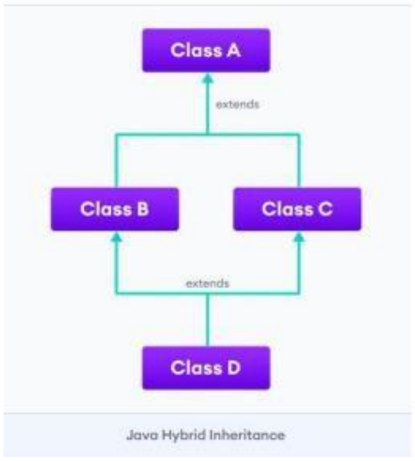


Multiple inheritance is not possible in JAVA. The reason behind this is to prevent ambiguity. Consider a case where class B extends class A and class C and both class A and C have the same method display (). Now javacompiler cannot decide, which display method it should inherit. To prevent such situation, multiple inheritances is not allowed in java.

➢ Hybrid Inheritance:

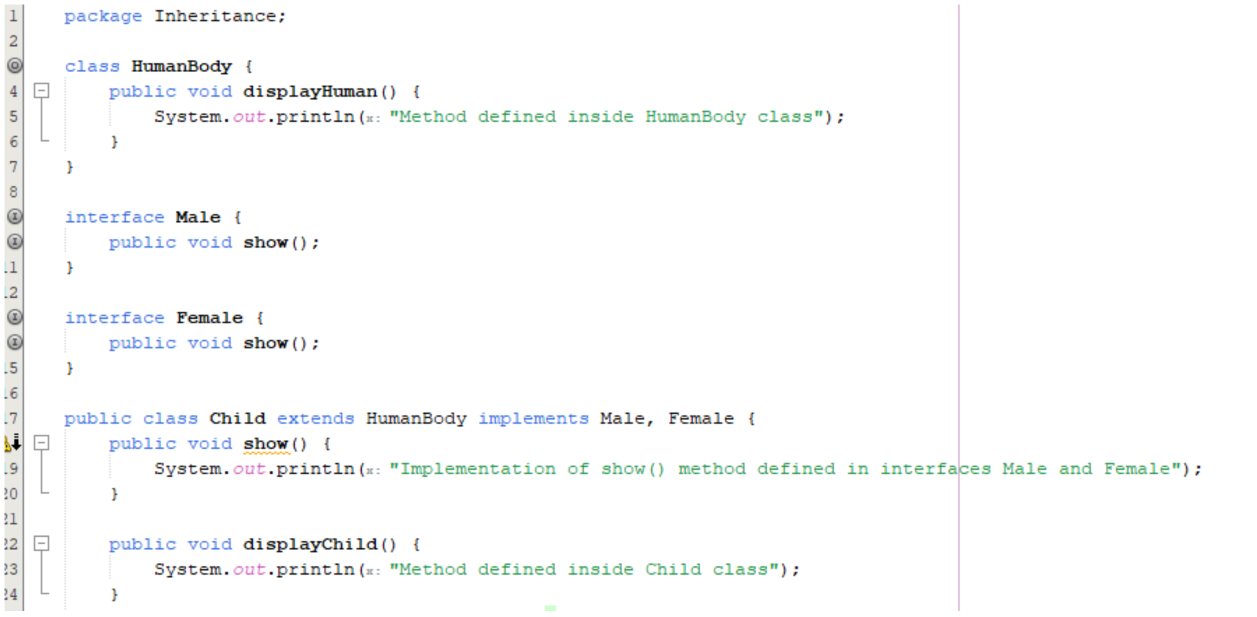
Hybrid inheritance is a combination of two or more types of inheritance.

- **For example:**

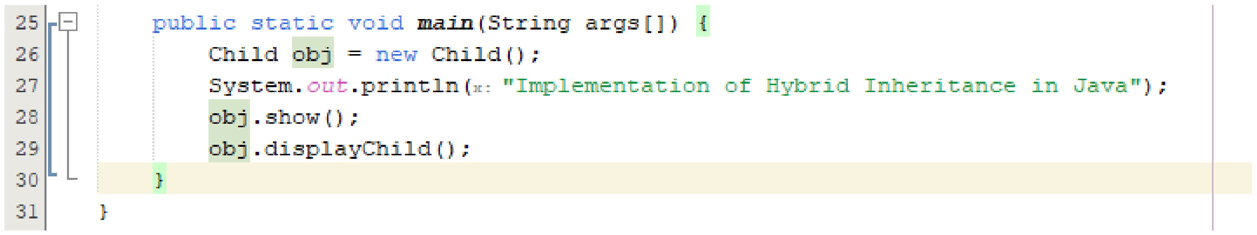


Java Hybrid Inheritance

- **Implementation of Hybrid Inheritance:**

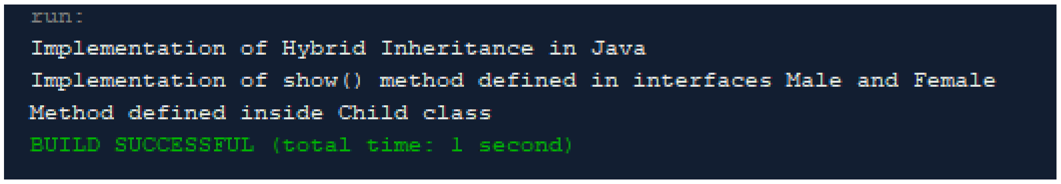


```java
package Inheritance;

class HumanBody {
    public void displayHuman() {
        System.out.println(x: "Method defined inside HumanBody class");
    }
}

interface Male {
    public void show();
}

interface Female {
    public void show();
}

public class Child extends HumanBody implements Male, Female {
    public void show() {
        System.out.println(x: "Implementation of show() method defined in interfaces Male and Female");
    }

    public void displayChild() {
        System.out.println(x: "Method defined inside Child class");
    }
```



```java
    public static void main(String args[]) {
        Child obj = new Child();
        System.out.println(x: "Implementation of Hybrid Inheritance in Java");
        obj.show();
        obj.displayChild();
    }
}
```

- **Output:**



```
run:
Implementation of Hybrid Inheritance in Java
Implementation of show() method defined in interfaces Male and Female
Method defined inside Child class
BUILD SUCCESSFUL (total time: 1 second)
```

Method Overloading and Overriding

➢ Method Overloading in JAVA:

If a class has multiple methods having the same name but different in parameters, it is known as MethodOverloading.

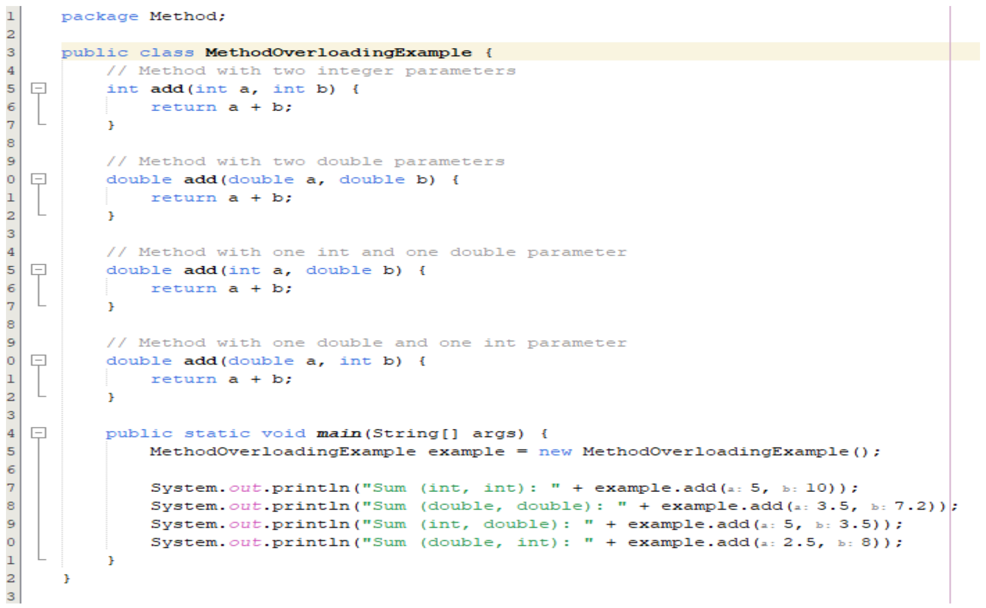
➢ Different ways to overload the method:

There are two ways to overload the method in Java

- By changing the number of arguments
- By changing the data type

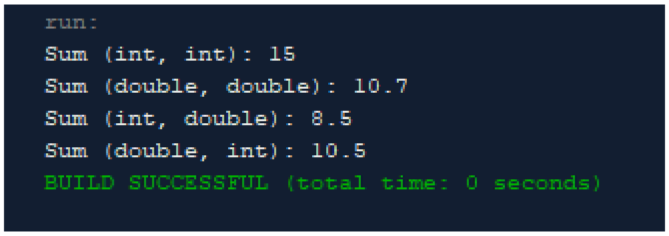
➢ Implementation of Method Overloading by changing the data type of arguments:

In this example, we have created two methods that differ in data type. The first display method receivesone integer arguments and second display method receives string arguments.



```java
package Method;

public class MethodOverloadingExample {
    // Method with two integer parameters
    int add(int a, int b) {
        return a + b;
    }

    // Method with two double parameters
    double add(double a, double b) {
        return a + b;
    }

    // Method with one int and one double parameter
    double add(int a, double b) {
        return a + b;
    }

    // Method with one double and one int parameter
    double add(double a, int b) {
        return a + b;
    }

    public static void main(String[] args) {
        MethodOverloadingExample example = new MethodOverloadingExample();

        System.out.println("Sum (int, int): " + example.add(a: 5, b: 10));
        System.out.println("Sum (double, double): " + example.add(a: 3.5, b: 7.2));
        System.out.println("Sum (int, double): " + example.add(a: 5, b: 3.5));
        System.out.println("Sum (double, int): " + example.add(a: 2.5, b: 8));
    }
}
```

➢ Output:



```
run:
Sum (int, int): 15
Sum (double, double): 10.7
Sum (int, double): 8.5
Sum (double, int): 10.5
BUILD SUCCESSFUL (total time: 0 seconds)
```

➢ Can we overload java main () method:

Yes, we can overload java main method by method overloading. We can have any number of main methodsin a class by method overloading. But JVM calls main () method which receives string array as arguments only.

➢ Method Overriding in Java:

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.

➢ Usage of Java Method Overriding:

• Method overriding is used to provide the specific implementation of a method which is already providedby its superclass.

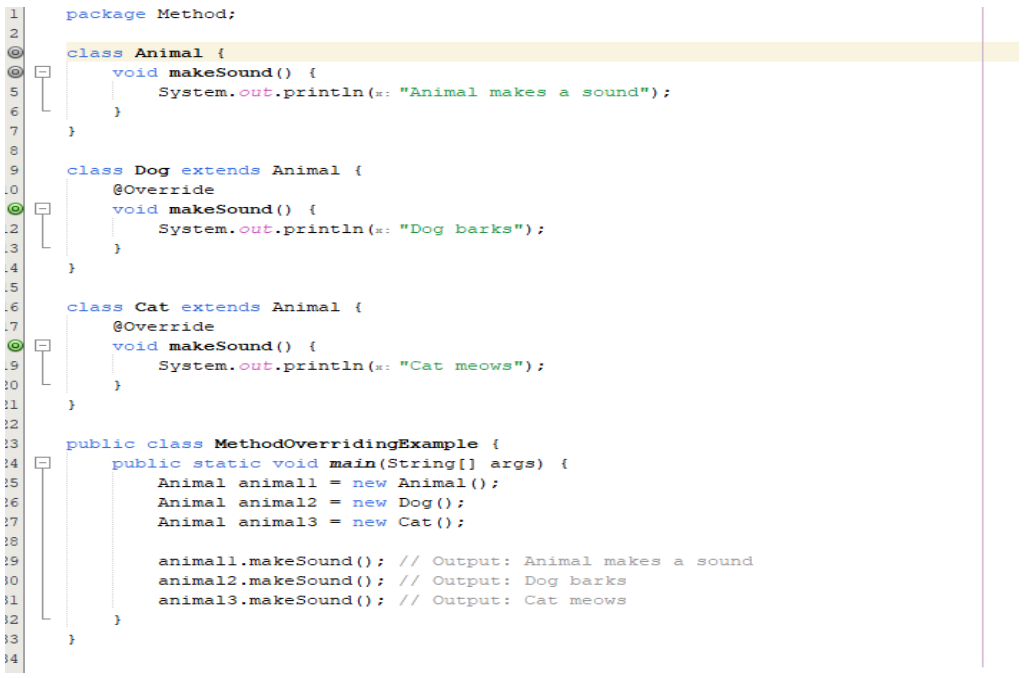
• Method overriding is used for runtime polymorphism.

➢ Rules for Java Method Overriding:

- The method must have the same name as in the parent class.
- The method must have the same parameter as in the parent class.
- There must be an IS-A relationship (inheritance).

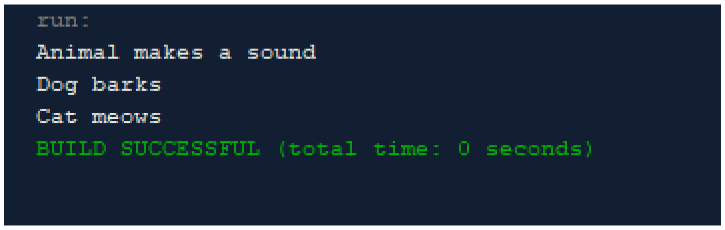
➢ Implementation of Method Overriding:

In this example, we have defined the display method in the subclass as defined in the parent class but it has some specific implementation. The name and parameter of the method are the same, and there is IS-A relationship between the classes, so there is method overriding.



```
package Method;

class Animal {
    void makeSound() {
        System.out.println(x: "Animal makes a sound");
    }
}

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

class Cat extends Animal {
    @Override
    void makeSound() {
        System.out.println(x: "Cat meows");
    }
}

public class MethodOverridingExample {
    public static void main(String[] args) {
        Animal animal1 = new Animal();
        Animal animal2 = new Dog();
        Animal animal3 = new Cat();

        animal1.makeSound(); // Output: Animal makes a sound
        animal2.makeSound(); // Output: Dog barks
        animal3.makeSound(); // Output: Cat meows
    }
}
```

➢ **Output:**



```
run:
Animal makes a sound
Dog barks
Cat meows
BUILD SUCCESSFUL (total time: 0 seconds)
```