[01]

class Example {

    public static void main(String[] args) {

        Student s1; //Create reference Variable for type "Student"

        s1 = new Student(); // Create an Object "Student" and initialize to s1

        s1.id = "S001";

        s1.name = "Student 1";

        s1.sub1 = 85;

        s1.sub2 = 80;

        System.out.println("id : " + s1.id);

        System.out.println("name : " + s1.name);

        System.out.println("sub1 : " + s1.sub1);

        System.out.println("sub2 : " + s1.sub2);

    }

}

[02] Attributes

class Student{

    // ----- Start Attribute Declaration

    String id;

    String name;

    int sub1;

    int sub2;

    // ----- End Attribute Declaration

}

[04]Constructors

class Box {

    int length;

    int width;

    int height;

    Box() {

        this.length = 1;

        this.width = 1;

        this.height = 1;

    }

    Box(int length, int width, int height) {

        this.length = length;

        this.width = width;

        this.height = height;

    }

[03] Behavior

 // ------ Start Method Declaration

    public void setValues(String stuId, String stuName, int stuSub1, int stuSub2){

        id = stuId;

        name = stuName;

        sub1 = stuSub1;

        sub2 = stuSub2;

    }

 public void printStudent(){

        System.out.println(id +", " + name + ", " + sub1 + ", " + sub2);

    }

    // ------ End Method Declaration

\*\*Inside the main method

 s1 = new Student(); // Create an Object "Student" and initialize to s1

 s1.setValues("S001", "Student 1", 85, 80);

s1.printStudent();

[06] Pass array to in a method

public class Main {

// Method to print elements of an array

public static void printArray(int[] arr) {

for (int num : arr) {

System.out.print(num + " ");

}

System.out.println();

}

public static void main(String[] args) {

int[] numbers = {1, 2, 3, 4, 5};

// Call the method and pass the array as an argument

printArray(numbers);

}

}

[05] Passing Object to Method

class Account{

    double balance;

    Account(double balance){

        this.balance = balance;

    }

    public void printBalance(){

        System.out.println("Balance is : " + this.balance);

    }

}

class Operation{

    public void withdraw(Account a1, double amount){

        a1.balance -= amount;

    }

    public void deposit(Account a1, double amount){

        a1.balance += amount;

    }

}

class Example {

    public static void main(String[] args) {

        Account a1 = new Account(10000);

        a1.printBalance();

        Operation operation = new Operation();

        operation.withdraw(a1, 5000);

        a1.printBalance();

        operation.deposit(a1, 7000);

        a1.printBalance();

    }

}

[7]

public class VarargsExample {

// Method that takes variable length arguments

public static void printNumbers(int... numbers) {

for (int number : numbers) {

System.out.print(number + " ");

}

System.out.println();

}

public static void main(String[] args) {

printNumbers(1, 2, 3); // Output: 1 2 3

printNumbers(4, 5); // Output: 4 5

printNumbers(6, 7, 8, 9, 10); // Output: 6 7 8 9 10

}

}

109]

class Animal {

public Animal(String name) {

this.name = name;

}

}

class Dog extends Animal {

public Dog(String name) {

super(name); // Calls the superclass constructor

}

}

[8]

// Superclass

class Animal {

String name;

public void eat() {

System.out.println(name + " is eating.");

}

}

// Subclass

class Dog extends Animal {

public void bark() {

System.out.println(name + " is barking.");

}

}

public class Main {

public static void main(String[] args) {

Dog dog = new Dog();

dog.name = "Buddy";

dog.eat(); // Inherited method

dog.bark(); // Subclass-specific method

}

}

[9]

class Dog extends Animal {

public void display() {

super.eat(); // Calls eat() method of superclass

}

}

[11]

class Animal {

public void sound() {

System.out.println("Animal makes a sound");

}

}

class Dog extends Animal {

@Override

public void sound() {

System.out.println("Dog barks");

}

}

[12]

abstract class Animal {

String name;

Animal(String name) {

this.name = name;

}

abstract void makeSound(); // Abstract method

void sleep() {

System.out.println(name + " is sleeping.");

}

}

class Dog extends Animal {

Dog(String name) {

super(name);

}

@Override

void makeSound() {

System.out.println(name + " says: Bark");

}

}

class Cat extends Animal {

Cat(String name) {

super(name);

}

@Override

void makeSound() {

System.out.println(name + " says: Meow");

}

}

public class Main {

public static void main(String[] args) {

Dog dog = new Dog("Buddy");

Cat cat = new Cat("Whiskers");

dog.makeSound(); // Outputs: Buddy says: Bark

dog.sleep(); // Outputs: Buddy is sleeping.

cat.makeSound(); // Outputs: Whiskers says: Meow

cat.sleep(); // Outputs: Whiskers is sleeping.

}

}

[13]

interface Animal {

void makeSound(); // Abstract method

}

interface Pet {

void play();

}

class Dog implements Animal, Pet {

public void makeSound() {

System.out.println("Bark");

}

public void play() {

System.out.println("Playing fetch");

}

}

class Cat implements Animal, Pet {

public void makeSound() {

System.out.println("Meow");

}

public void play() {

System.out.println("Playing with a ball of yarn");

}

}

public class Main {

public static void main(String[] args) {

Dog dog = new Dog();

Cat cat = new Cat();

dog.makeSound(); // Outputs: Bark

dog.play(); // Outputs: Playing fetch

cat.makeSound(); // Outputs: Meow

cat.play(); // Outputs: Playing with a ball of yarn

}

}

[14]

@FunctionalInterface

interface Add {

int add(int a, int b);

}

public class LambdaExample {

public static void main(String[] args) {

Add addition = (a, b) -> a + b;

System.out.println("Sum: " + addition.add(5, 3)); // Output: Sum: 8

}

}

[15]

import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
  
public class DBConnection {  
 private static DBConnection dBConnection;  
 private Connection connection;  
   
 private DBConnection() throws ClassNotFoundException, SQLException{  
 Class.forName("com.mysql.cj.jdbc.Driver");  
 connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/project1", "username", "password");  
 }  
   
 public static DBConnection getInstance() throws ClassNotFoundException, SQLException{  
 if(dBConnection == null){  
 dBConnection = new DBConnection();  
 }  
 return dBConnection;  
 }  
   
 public Connection getConnection(){  
 return connection;  
 }  
}

[16]

class Animal {

public void makeSound() {

System.out.println("Animal sound");

}

}

class Dog extends Animal {

public void makeSound() {

System.out.println("Bark");

}

}

public class UpcastingExample {

public static void main(String[] args) {

Dog dog = new Dog();

Animal animal = dog; // Implicit upcasting

animal.makeSound(); // Output: Bark }}

[17]

public class DowncastingExample {

public static void main(String[] args) {

Animal animal = new Dog(); // Upcasting

Dog dog = (Dog) animal; // Explicit downcasting

dog.makeSound(); // Output: Bark

Animal animal2 = new Animal();

if (animal2 instanceof Dog) {

Dog dog2 = (Dog) animal2; // Safe downcasting

dog2.makeSound();

} else {

System.out.println("animal2 is not an instance of Dog");

}

}

}

[18]

Animal animal = new Dog();

if (animal instanceof Dog) {

Dog dog = (Dog) animal;

dog.makeSound(); // Output: Bark

} else {

System.out.println("The object is not an instance of Dog");

}