TP4-SOLUTION

Object-oriented programming

(inheritance)

Exercise 01 :

```
class Shape {
    private String color;
    private boolean filled;

    public Shape() {
        color = "red";
        filled = true;
    }

    public Shape(String color, boolean filled) {
        this.color = color;
        this.filled = filled;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public boolean isFilled() {
        return filled;
```

```
    }

    public void setFilled(boolean filled) {
        this.filled = filled;
    }

    public String toString() {
        return "Shape[color=" + color + ", filled=" + filled + "]";
    }
}

class Circle extends Shape {
    private double radius;

    public Circle() {

        radius = 1.0;
    }

    public Circle(double radius,String color, boolean fill) {
        super(color,fill);
        this.radius = radius;
    }

    public double getRadius() {
        return radius;
    }

    public void setRadius(double radius) {
        this.radius = radius;
```

```
    }

    @Override
    public String toString() {
        return "Circle[Shape[color=" + getColor() + ", filled=" + isFilled() + "], radius=" + radius + "]";
    }
}

class Rectangle extends Shape {
    private double width;
    private double length;

    public Rectangle() {
        width = 1.0;
        length = 1.0;
    }

    public Rectangle(double width, double length) {
        this.width = width;
        this.length = length;
    }

    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        this.width = width;
    }
```

```
    public double getLength() {
        return length;
    }

    public void setLength(double length) {
        this.length = length;
    }

    @Override
    public String toString() {
        return "Rectangle[Shape[color=" + getColor() + ", filled=" + isFilled() + "], width=" + width + ",
length=" + length + "]";
    }
}
```

Exercise 02:

```
// Parent class Computer
class Computer {
    protected String brand;

    public Computer(String brand) {
        this.brand = brand;
    }

    public void boot() {
        System.out.println("Starting the computer...");
    }

    public void shutdown() {
        System.out.println("Shutting down the computer...");
    }
```

```
}

// Child class Laptop inheriting from Computer
class Laptop extends Computer {
    private double screenSize;

    public Laptop(String brand, double screenSize) {
        super(brand);
        this.screenSize = screenSize;
    }

    public void sleepMode() {
        System.out.println("Sleep mode for the laptop...");
    }
}

// Child class SmartPhone inheriting from Computer
class SmartPhone extends Computer {
    private String operatingSystem;

    public SmartPhone(String brand, String operatingSystem) {
        super(brand);
        this.operatingSystem = operatingSystem;
    }

    public void call() {
        System.out.println("Making a call from the smartphone...");
    }
}
```

```
// Main class to demonstrate inheritance
public class Main {
    public static void main(String[] args) {
        // Create instances of Laptop and SmartPhone
        Laptop laptop = new Laptop("Dell", 15.6);
        SmartPhone phone = new SmartPhone("Apple", "iOS");

        // Demonstrate methods of Laptop and SmartPhone
        laptop.boot();      // Output: Starting the computer...
        laptop.sleepMode();  // Output: Sleep mode for the laptop...
        phone.boot();       // Output: Starting the computer...
        phone.call();       // Output: Making a call from the smartphone...
    }
}
```

Exercise 03:

```
// Person.java
// Parent class Person

public class Person {
    private String firstName;
    private String lastName;

    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public String getFirstName() {
        return firstName;
```

```
    }

    public String getLastName() {
        return lastName;
    }
}
// Employee.java
// Child class Employee
public class Employee extends Person {
    private int employeeId;
    private String jobTitle;

    public Employee(String firstName, String lastName, int employeeId, String jobTitle) {
        super(firstName, lastName);
        this.employeeId = employeeId;
        this.jobTitle = jobTitle;
    }

    public int getEmployeeId() {
        return employeeId;
    }

    @Override
    public String getLastName() {
        return super.getLastName() + ", " + jobTitle;
    }
}
// Main.java
// Main class
public class Main {
```

```
    public static void main(String[] args) {
     Employee employee1 = new Employee("Kortney", "Rosalee", 4451, "HR Manager");
    System.out.println(employee1.getFirstName() + " " + employee1.getLastName() + " (" +
employee1.getEmployeeId() + ")");
            Employee employee2 = new Employee("Junior", "Philipa", 4452, "Software Manager");
    System.out.println(employee2.getFirstName() + " " + employee2.getLastName() + " (" +
employee2.getEmployeeId() + ")");
    }
}
```

Exercise 04:

Write a Java program to create a class known as "BankAccount" with methods called deposit() and withdraw(). Create a subclass called SavingsAccount that overrides the withdraw() method to prevent withdrawals if the account balance falls below one hundred.

```
// BankAccount.java
// Parent class BankAccount
import java.util.Random;
public class BankAccount {
   private String accountNumber;
   private double balance;

   public BankAccount(double balance) {
       this.accountNumber = generateAccountNumber();
      this.balance = balance;
   }
public String getAccountNumber() {
      return accountNumber;
   }
private String generateAccountNumber() {
```

```
        Random random = new Random();
        int number = random.nextInt(90000000) + 10000000; // Generates a random number between 10000000 and 99999999
        return String.format("%08d", number); // Formats the number as 8 digits with leading zeros if necessary
    }
    public void deposit(double amount) {
        balance += amount;
    }
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
        } else {
            System.out.println("Insufficient balance");
        }
    }

    public double getBalance() {
        return balance;
    }
}
```

// SavingsAccount.java

// Child class SavingsAccount

```
public class SavingsAccount extends BankAccount {
    public SavingsAccount(String accountNumber, double balance) {
        super(accountNumber, balance);
    }

    @Override
```

```
    public void withdraw(double amount) {
        if (getBalance() - amount < 100) {
            System.out.println("Minimum balance of $100 required!");
        } else {
            super.withdraw(amount);
        }
    }
}
// Main.java
// Main class
public class Main {
    public static void main(String[] args) {
 System.out.println("Create a Bank Account object with initial balance of $500:");
//Create a BankAccount object with initial balance of $500
BankAccount BA1234 = new BankAccount( 500);
// Deposit $1000 into account BA1234
System.out.println("Deposit $1000 into account BA1234:");
BA1234.deposit(1000);
 System.out.println("New balance after depositing $1000: $" + BA1234.getBalance());
// Withdraw $600 from account BA1234
System.out.println("Withdraw $600 from account BA1234:");
BA1234.withdraw(600);
System.out.println("New balance after withdrawing $600: $" + BA1234.getBalance());
// Create a SavingsAccount object with initial balance of $450
System.out.println("\nCreate a SavingsAccount object with initial balance of $450:");
SavingsAccount SA1234 = new SavingsAccount(450);
// Withdraw $300 from SA1234
SA1234.withdraw(300);
System.out.println("Balance after trying to withdraw $300: $" + SA1234.getBalance());
// Create a SavingsAccount object with initial balance of $300
```

```
System.out.println("\nCreate a SavingsAccount object with initial balance of $300:");
SavingsAccount SA1000 = new SavingsAccount(300);
// Withdraw $250 from SA1000 (balance falls below $100)
System.out.println("Try to withdraw $250 from SA1000!");
SA1000.withdraw(250);
  System.out.println("Balance after trying to withdraw $250: $" + SA1000.getBalance());

   }
}
```