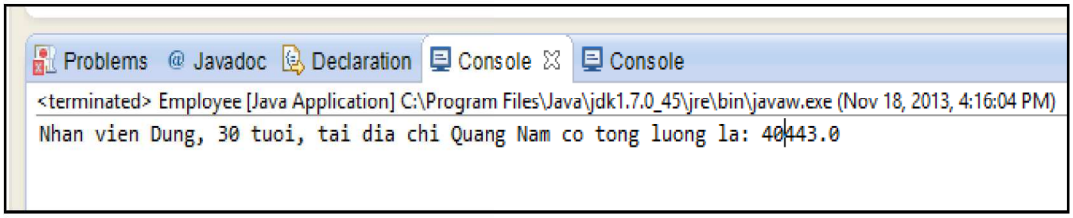
Chapter 6. Implementation of Inheritance and Polymorphism

I. Exercises with solutions

1. Write an OOP program, creating a class Person with attributes *name, age, adress* and methods: *constructors, attributes information display*. Create an class Employee inheriting the class Person. The class Employee has its own attributes *salary, rate* and methods *constructors, salary calculation and employee information display*.

```
/*Tạo lớp Person*/
class Person
{

/*Khai báo các thuộc tính cho lớp */
      private String name;
      private int age;
      private String address;

/*Phương thức khởi tạo*/
      public Person(String name, int age, String address)
      {

/*Khởi tạo giá trị cho các thuộc tính của đối tượng hiện đang
gọi phương thức này*/
            this.name=name;
            this.age =age;
            this.address =address;
      }

/*Hiển thị thông tin */
      public void display()
      {
      System.out.print("Nhan vien " + name+", "+age+" tuoi,
tai dia chi "+address);
      }
}

/*Tạo lớp Employee kế thừa lớp Person*/
public class Employee extends Person
{

/*Tạo thêm hai thuộc tính cho Employee*/
      private float salary;
      private float rate;

/*Định nghĩa hàm khởi tạo cho lớp Employee*/
  public Employee(String name, int age, String address, float
       salary, float rate)
```

```
      {

    /*Gọi hàm khởi tạo của lớp Person để khởi tạo thuộc tính*/
           super(name,age,address);

    /*Khởi tạo thuộc tính của Employee*/
          this.salary=salary;
          this.rate=rate;
      }

/*Định nghĩa hàm tính tổng lương*/
      public float totalSalary()
      {
          return salary*rate;
      }

/*Định nghĩa hàm hiển thị thông tin của Employee*/
      public void display()
      {

       /*Gọi hàm display của lớp Person để hiển thị name, age
và address*/
          super.display();

       /*Hiển thị thông tin lương*/
           System.out.print(" co tong luong la:
"+totalSalary());
      }
      public static void main(String[] args) {

       /*Tạo đối tượng của lớp Employee*/
       Employee A = new Employee("Dung", 30, "Quang
Nam",11050,3.66f);

       /*Hiển thị thông tin của A*/
      A.display();
      }
}
```



2. Write an OOP program, creating the following classes:

+ an abstract class Shape

+ a class Rectangle, that extends Shape, describes a simple rectangle.

+ a class Circle, that extends Shape, describes a simple circle.

The class Shape class declares an abstract method of area. The classes Rectangle and Circle implement the abstract method of area inherited from class Shape.

```
abstract class Shape
{
  public abstract double area();
  public String toString()
    {
     return "The area is " + area();
    }
}
class Rectangle extends Shape
{
   private double width, height;
   public Rectangle(double wVal, double hVal)
    {
     width = wVal;
     height = hVal;
    }
   public double area()
    {
    return width*height;
    }
}
class Circle extends Shape
{
   private double radius;
   public Circle(double rad)
    {
    radius = rad;
    }
   public double area()
    {
    return Math.PI * radius * radius;
    }
}
public class AbstractShape
{
   public static void main(String args[])
    {
    Shape vec[] = {new Circle(5), new Rectangle(4,5), new
Circle(4), new Rectangle(7,8)};
    for(int index = 0; index < vec.length; index ++)
```

```
        {
        System.out.println(vec[index]);
        }
      }
}
```

3. Write an OOP program, creating the following classes:

+ an abstract class Employee

+ a Manager class, that extends Employee, describes a manager.

+ a Clerk class, that extends Employee, describes a clerk.

The class Employee class declares an abstract method of salary. The classes Manager and Clerk implement the abstract method of salary inherited from class Employee.

```
abstract class Employee
{
   private String name;
   private double age;
   private double hourRate;
   public Employee(String name, double age, double hourRate)
      {
      this.name = name;
      this.age = age;
      this.hourRate = hourRate;
      }

   public abstract double salary(double hours);

   public String getName()
   {
     return name;
   }
   public double getHourRate()
   {
     return hourRate;
   }
   public String toString()
    {
    return "name= " + name + " age=" + age + " hourRate=" +
           hourRate;
    }
}
class Manager extends Employee
{
  public  Manager(String name, double age, double hourRate)
```

```
    {
     super(name, age, hourRate);
    }
  public double salary(double hours)
    {
     return hours * this.getHourRate() * 1.2;
    }
}
class Clerk extends Employee
{
   public Clerk(String name, double age, double hourRate)
    {
     super(name, age, hourRate);
    }
   public double salary(double hours)
    {
    return hours * this.getHourRate() * 0.8;
    }
}
public class AbstractEmployee
{
   public static void main(String args[])
     {
     Employee vec[] = {new Manager("HCPhap", 45, 104), new
Clerk("Jonh Le", 26, 110), new Manager("David Huynh", 42,
120), new Manager("Marry Nguyen", 34, 120)};
     double hours[] = {140,150,130,180};
     double total = 0;
     for(int index = 0; index < vec.length; index ++)
      {
      total += vec[index].salary(hours[index]);
      System.out.println(vec[index]);
      }
     System.out.println("The total payment of the employees is
" + total);
    }
}
```

```
name= HCPhap age=45.0 hourRate=104.0
name= Jonh Le age=26.0 hourRate=110.0
name= David Huynh age=42.0 hourRate=120.0
name= Marry Nguyen age=34.0 hourRate=120.0
The total payment of the employees is 75312.0
```

4. Write an OOP program, creating 10 objects of the following classes: Rectangle, Circle and EquilateralTriangle. Each of these classes should extend the abstract class

Shape in the previous exercise. Students should declare the three classes and write code to sort 10 created objects according to their area.

```
import java.util.Arrays;
abstract class Shape implements Comparable
{
  abstract double area();

  /*implement method compareTo of class Comparable
   * in order to compare two objects according to their area
   */
  public int compareTo(Object other)
    {
    return (int)(this.area()-((Shape)other).area());
    }
  public String toString()
    {
    return "area="+area();
    }
}
class Circle extends Shape
{
    double radius;
    Circle(double radiusVal)
      {
      radius = radiusVal;
      }
    double area()
     {
      return Math.PI * Math.pow(radius,2);
     }
    public String toString()
     {
      return "Circle:"+super.toString();
     }
}
class Rectangle extends Shape
{
    double width, height;
    Rectangle(double widthVal, double heightVal)
    {
     width = widthVal;
     height = heightVal;
    }
   double area()
    {
    return width*height;
    }
   public String toString()
```

```
    {
    return "Rectangle:"+super.toString();
    }
}
class EquilateralTriangle extends Shape
{
    double length;
    EquilateralTriangle(double length)
     {
     this.length = length;
     }
    double area()
     {
     return 0.5*length*Math.sin((2*60*Math.PI/360));
     }
    public String toString()
     {
     return "EquilateralTriangle:"+super.toString();
     }
}
public class ShapeSort
{
   public static void main(String args[])
     {
     Shape vec[] = new Shape[10];

     //automatically initialize values for objects
     for(int index=0; index<vec.length; index++)
      {
      switch((int)(4*Math.random()))
        {
        case 1:
         vec[index] = new Rectangle(1000*Math.random(),
1000*Math.random());
         break;
        case 2:
          vec[index] = new Circle(1000*Math.random());
         break;
        default:
         vec[index] = new
EquilateralTriangle(1000*Math.random());
        }
    }

     Arrays.sort(vec);

        for(int index=0; index<vec.length; index++)
        {
          System.out.println(vec[index]);
        }
```

```
    }
}
```

```
Rectangle:area=158685.0192807511
Circle:area=292793.5868513119
Rectangle:area=569295.2494713358
Circle:area=601084.1846221576
Circle:area=770995.0773476631
Circle:area=928398.7038432377
```

5. Write an OOP program, creating an interface Vehicle declaring abstract methods. Creating a class Motobike and a class Car implementing the interface Vehicle by overriding all abstract inherited methods.

```
interface Vehicle
{
      // all are the abstract methods.
      public void changeGear(int a);
      public void speedUp(int a);
      public void brake(int a);
}

class MotoBike implements Vehicle{

      private String brand;
      private int speed;
      private int gear;
      private int wheels;

      public MotoBike(String brand,int speed, int gear, int
wheels)
      {
       this.brand=brand;
       this.speed=speed;
       this.gear=gear;
       this.wheels=wheels;
      }
      // to change gear, overide
      public void changeGear(int newGear)
      {
       gear = newGear;
      }

      // to increase speed, Override
      public void speedUp(int increment)
      {
       speed = speed + increment;
```

```
    }

    // to decrease speed, Override
    public void brake(int decrement)
    {
     speed = speed - decrement;
    }

    public void printStates() {
     System.out.println("Brand: "+brand+", wheels:
"+wheels+", speed: " + speed + ", gear: " + gear);
    }
}

class Car implements Vehicle {

    private String brand;
    private int speed;
    private int gear;
    private int wheels;

    public Car(String brand,int speed, int gear, int
wheels)
    {
        this.brand=brand;
        this.speed=speed;
        this.gear=gear;
        this.wheels=wheels;
    }

    // to change gear, Override
    public void changeGear(int newGear)
    {
        gear = newGear;
    }
    // to increase speed, Override
    public void speedUp(int increment)
    {
        speed = speed + increment;
    }

    // to decrease speed, Override
    public void brake(int decrement)
    {
        speed = speed - decrement;
    }

    public void printStates() {
        System.out.println("Brand: "+brand+", wheels:
"+wheels+", speed: " + speed + ", gear: " + gear);
```

```
    }

}
class VehicleInterface {

    public static void main (String[] args) {

        // creating an inatance of Bicycle
        // doing some operations
        MotoBike bike = new MotoBike("Honda",100,4,2);
        bike.changeGear(2);
        bike.speedUp(3);
        bike.brake(1);

        System.out.println("Motobike:");
        bike.printStates();

        // creating instance of the bike.
        Car car = new Car("Mitsubishi",300,6,4);
        car.changeGear(1);
        car.speedUp(4);
        car.brake(3);

        System.out.println("Car:");
        car.printStates();
    }
}
```

```
Motobike:
Brand: Honda, wheels: 2, speed: 102, gear: 2
Car:
Brand: Mitsubishi, wheels: 4, speed: 301, gear: 1
```

6. Write an OOP program, creating an interface Message declaring abstract methods. Creating a class TextMessage and a class BinaryMessage implementing the interface Message by overriding all abstract inherited methods.

```
interface Message
{
    public String getHeader();
    public void    setHeader(String str);

    public String getBody();
```

```
      public void    setBody(String str);

      public String getType();
      public void    setType(String str);
      public void printMessage();
}
class TextMessage implements Message
{
      String header;
      String body;
      String type;
      public TextMessage(String header, String body)
      {
       this.body=body;
       this.header =header;
       this.type="text";
      }
      public String getHeader()
      {
       return header;
      }
      public void    setHeader(String str)
      {
       header=str;
      }

      public String getBody()
      {
       return body;
      }
      public void    setBody(String str)
      {
      body=str;
      }

      public String getType()
      {
       return type;
      }
      public void    setType(String str)
      {
       type=str;
      }
      public void printMessage()
      {
      System.out.println("Type: "+type+"\nheader:
"+header+"\nbody: "+body);
      }
      }
class BinaryMessage implements Message
```

```
{
      String header;
      String body;
      String type;
      public BinaryMessage(String header, String body)
      {
       this.body=body;
       this.header =header;
       this.type="binary";
      }
      public String getHeader()
      {
       return header;
      }
      public void    setHeader(String str)
      {
       header=str;
      }

      public String getBody()
      {
       return body;
      }
      public void    setBody(String str)
      {
      body=str;
      }

      public String getType()
      {
       return type;
      }
      public void    setType(String str)
      {
       type=str;
      }
      public void printMessage()
      {
      System.out.println("Type: "+type+"\nheader:
"+header+"\nbody: "+body);
      }
}


class MessageInterface {

      public static void main (String[] args) {
            TextMessage msg1 = new
TextMessage("time2021:oganizerVKU:createrHCPhap","This is an
text message created by HCPhap-VKU");
```

```
            BinaryMessage msg2 = new
BinaryMessage("time2021:oganizerVKU:createrHCPhap","10101010
10101010111110101");

            msg1.printMessage();
            msg2.printMessage();
      }
}
```

```
Type: text
header: time2021:oganizerVKU:createrHCPhap
body: This is an text message created by HCPhap-VKU
Type: binary
header: time2021:oganizerVKU:createrHCPhap
body: 10101010101010111110101
```

7. Write an OOP program of students management, including basic functionalities as follows:

- Input a list of students,
- Display information of the list of students
- Sort the students list according to their average points,
- Searh a student by name.

```
import java.util.Scanner;
import java.util.Vector;
import java.util.Enumeration;
import java.util.Arrays;

public class StudentManagement
{

/*dung de chua danh sach sinh vien*/
 Vector list = new Vector();

 public StudentManagement()
 {
  while(true)
  {

/*Hien thi menu chuong trinh*/
   System.out.println("*-CHUONG TRINH QUAN LY SINH VIEN-*");
   System.out.println("*-Chuc nang chinh chuong trinh-*");
   System.out.println("  1. Nhap danh sach sinh vien   ");
   System.out.println("  2. Xem danh sach sinh vien   ");
   System.out.println("  3. Sap xep sach sinh vien tang dan
diem trung binh   ");
```

```
   System.out.println("  4. Tim sinh vien theo ten   ");
   System.out.println("  5. Thoat   ");
   System.out.println("  -------------");

 /*Nhập một số từ bàn phím*/
   int num;
   Scanner keyboard = new Scanner(System.in);
   System.out.print("  Nhap mot so de chon chuc nang: ");
   num = keyboard.nextInt();

/*Kiểm tra và gọi chức năng tương ứng*/
   switch(num)
   {
   case 1:
        this.input();
        break;
   case 2:
        this.view();
        break;
   case 3:
        sort();
        break;
   case 4:
        search();
        break;
   case 5:
        System.out.print("----  Chuong trinh ket thuc----- ");
        return;
        }
  }
 }

/*Nhap danh sach sinh vien*/
   public void input()
   {

   /*Nhập số lượng sinh viên cho danh sách*/
      int num;
      Scanner keyboard = new Scanner(System.in);
      System.out.print("  Nhap so luong sinh vien: ");
      num = keyboard.nextInt();

   /*Nhập thông tin cho mỗi sinh viên*/
      for (int i=1;i<=num;i++)
      {
      System.out.println("  Nhap thong tin cho sinh vien thu:
"+i);
      System.out.print("  ID: ");
      int id = Integer.parseInt(keyboard.next());
```

```
      keyboard.nextLine();//xoa bo dem
      System.out.print("  Ten: ");
      String name = keyboard.nextLine();

      System.out.print("  Diem trung binh: ");
      float aver = keyboard.nextFloat();

  /*Sau khi nhập thông tin, tạo đối tượng sinh viên*/
      Student st = new Student(id,name,aver);

  /*Lưu đối tượng sinh viên vào danh sách*/
         list.add(st);
     }
     System.out.println("\n--------\n");
    }

/*Xem danh sach sinh vien*/
    public void view()
    {

    /*Hiển thị danh sách sinh viên*/
     System.out.println("  Thong tin danh sach sinh vien");

    /*Lấy sinh viên từ danh sách (vector) và lưu trữ ở vEnum*/
     Enumeration vEnum = list.elements();

    /*Duyệt từng phần tử của vEnum*/
     int i=1;

    /*Chưa hết phần tử*/
     while(vEnum.hasMoreElements())
      {

     /*Lấy phần tử từ vEnum ép lại kiểu Student*/
      Student sts = (Student)vEnum.nextElement();

     /*Hiển thị thông tin sinh viên*/
      System.out.println("    "+i+". ID="+sts.getId()+",
Ten="+sts.getName()+", Diem trung binh="+sts.getAver());
         i++;
      }
       System.out.println("\n--------\n");
    }

/*sap xep danh sach theo chieu tang dan cua diem trung binh
su dung ham sort cuar lop Arrays*/
    public void sort()
    {
```

```
   /*Đổ dữ liệu từ Vector vào mảng để gọi hàm sort sắp xếp
mảng*/
     Student[] sts = new Student[list.size()];
     int index=0;

     Enumeration vEnum = list.elements();
     while(vEnum.hasMoreElements())
      {
       sts[index] = (Student)vEnum.nextElement();
       index++;
      }

    /*Sắp xếp mảng*/
      Arrays.sort(sts);

       System.out.println("\n--Danh sach sinh vien sau khi sap
xep--");
     for(index=0; index < sts.length; index++)
        {

    /*Hiển thị thông tin sinh viên sau khi sắp xếp*/
       System.out.println("    "+(index+1)+".
ID="+sts[index].getId()+", Ten="+sts[index].getName()+", Diem
trung binh="+sts[index].getAver());
        }
       System.out.println("\n--------\n");

        }

/*Tìm kiếm sinh viên theo tên*/
   public void search()
   {

    /*Nhập tên sinh viên cần tìm kiếm*/
      Scanner keyboard = new Scanner(System.in);
      System.out.print(" Nhap ten sinh vien can tim: ");
      String name = keyboard.nextLine();

   /*Duyệt từng phần tử của mảng để so sánh tên tìm kiếm*/
      Enumeration vEnum = list.elements();

       System.out.println("\n--Thong tin tim kiem duoc--");
      while(vEnum.hasMoreElements())
      {
      Student sts = (Student)vEnum.nextElement();

       /*Nếu tên sinh viên chứa chuỗi nhập vào thì hiển thị
    thông tin đối     tượng sinh viên*/
```

```
      if (sts.getName().indexOf(name)!=-1)
        System.out.println("ID="+sts.getId()+",
Ten="+sts.getName()+", Diem trung binh="+sts.getAver());
      }
        System.out.println("\n--------\n");

   }

 public static void main(String[] args)
   {
  /*Khởi tạo chương trình*/
    new StudentManagement();
   }
}

/*Xây dựng lớp Student hiện thực interface Comparable, định
nghĩa cụ thể hàm compareTo để phục vụ sắp xếp*/



class Student implements Comparable
{
      private int id;
      private String name;
      private float aver;

/*Hàm khởi tạo mặc định*/
      public Student()
      {
            name = new String("");
            id = 0;
            aver=0;
      }

/*Hàm khởi tạo 3 đối số*/
      public Student(int i, String n, float a)
      {
            id = i;
            name = n;
            aver=a;
      }

/*Hàm lấy giá trị name*/
      public String getName()
      {
            return name;
      }

/*Hàm lấy giá trị id*/
```

```
	public int getId()
	{
		return id;
	}

/*Hàm lấy giá trị Aver*/
	public float getAver()
	{
		return aver;
	}

/*Hàm so sánh 2 đối tượng Student phục vụ sắp xếp*/
	public int compareTo(Object other)
	   {
	    Student otherRect = (Student)other;
	    return (int)(this.aver-otherRect.aver);
	   }
}
```

```
Problems  @ Javadoc  Declaration  Console  Console
<terminated> StudentManagement (1) [Java Application] C:\Program Files\Java
*-CHUONG TRINH QUAN LY SINH VIEN-*
*-Chuc nang chinh chuong trinh-*
  1. Nhap danh sach sinh vien
  2. Xem danh sach sinh vien
  3. Sap xep sach sinh vien tang dan diem trung binh
  4. Tim sinh vien theo ten
  5. Thoat
  -------------
  Nhap mot so de chon chuc nang: 1
  Nhap so luong sinh vien: 3
  Nhap thong tin cho sinh vien thu: 1
  ID: 1
  Ten: hcphap
  Diem trung binh: 8
  Nhap thong tin cho sinh vien thu: 2
  ID: 2
  Ten: ddtho
  Diem trung binh: 9
  Nhap thong tin cho sinh vien thu: 3
  ID: 3
  Ten: lmthai
  Diem trung binh: 9.5

 --------
```

II. Do it yourself

1. Write an OOP program, creating the following classes:

+ an abstract class Shape, declaring abstract methods: area, perimeter, draw.
+ a class Square, that extends Shape, describes a simple square.
+ a class Circle, that extends Shape, describes a simple circle.
+ a class EquilateralTriangle, that extends Shape, describes an equilateral triangle.

2. Write an OOP program, creating an interface Animal declaring abstract methods. Creating a class Dog and a class Cat implementing the interface Animal by overriding all abstract inherited methods.
3. Write an OOP program, reusing the classes TextMessage and BinaryMessage above, creating a class MessageProducer and a class MessageConsumer. The class MessageProducer includes methods *create, send* and other methods. The class MessageConsumer includes methods *receive, show* and other methods.
4. Run all exercises with solutions and work yourself all exercises in **Chapter 2 of the Book "Bài tập lập trình Java Cơ bản –** Có lời giải**"**