МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ

***Циклова комісія програмування***

**Робочий план**

з курсу «Java - програмування»

Практична робота №3

Солом’яного Ярослава Сергійовича

*ПІБ студента*

студента групи **2П-18**

|  |  |  |
| --- | --- | --- |
| **Вид роботи** | **Дата** | **Оцінка** |
| *Пром. контроль* |  |  |
| *Залік* |  |  |

Викладач Марченко С. В.

Черкаси-2021

Код GitHub: [Link](https://github.com/Yaroslav-Solomyaniy/StudyCSBC/tree/main/Java/Practice%203/Practice%203/src)

Завдання 1. Написати клас, у якому буде присутній узагальнений метод printArray для друку елементів масивів. Метод main() повинен містити три масиви для тестування:



Рис 1. Приклад введених масивів

Виконайте перевантаження (overloading) узагальненого методу printArray так, щоб неузагальнена версія методу друкувала конкретно масиви елементів типу String у наступному форматі

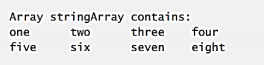


Рис 2. Приклад виводу масивів

Лістинг 1. Код для завдання 1.

package Ex\_1;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class Test  
{  
 public static <E> void printArray(E[] inputArray) {  
 for (E element : inputArray)  
 System.out.printf("%s ", element);  
 }  
  
 public static void main(String args[]) {  
 Integer[] integerArray = { 1, 2, 3, 4, 5, 6 };  
 Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 };  
 Character[] characterArray = { 'H', 'E', 'L', 'L', 'O' };  
 System.out.println("Array integerArray contains:");  
 printArray(integerArray);  
 System.out.println("\nArray doubleArray contains:");  
 printArray(doubleArray);  
 System.out.println("\nArray characterArray contains:");  
 printArray(characterArray);  
 printTimeStamp();  
 }  
 public static void printTimeStamp() {  
 final String sname = "Yaroslav Solomianyi";  
 Date currentDate = new Date();  
 SimpleDateFormat dateFormat = null;  
 dateFormat = new SimpleDateFormat();  
 System.out.println("\nMade by: " + sname + " on " + dateFormat.format(currentDate));  
 }  
}

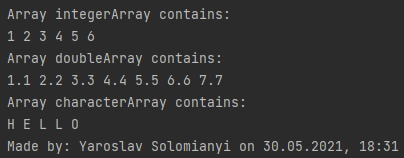


Рис 3. Результат виконанного завдання 1.

Завдання 2. Реалізувати інтерфейс Меблі, абстрактний клас Шафа та клас Книжкова шафа, а також наслідування та поліморфізм для них. Опишіть в інтерфейсі один або більше методів за замовчуванням.

Лістинг 2. Інтерфейс меблі

package Ex\_2;  
  
public interface Furniture {  
 void cost(double a,double b, double c);  
 void cost(double v, double v1, double v2, int i, int i1);  
}

Лістинг 3. Абстрактний клас Шафа

package Ex\_2;  
  
abstract class WardRobe implements Furniture{  
  
 protected double length;  
 protected double width;  
 protected double height;  
 {  
 length = 0;  
 width = 0;  
 height = 0;  
 }  
  
  
 public WardRobe(double length, double width, double height) {  
 this.length = length;  
 this.width = width;  
 this.height = height;  
 }  
  
 public double getLength() {  
 return length;  
 }  
 public void setLength(double length) {  
 this.length = length;  
 }  
 public double getWidth() {  
 return width;  
 }  
 public void setWidth(double width) {  
 this.width = width;  
 }  
 public double getHeight() {  
 return height;  
 }  
 public void setHeight(double height) {  
 this.height = height;  
 }  
  
 @Override  
 public String toString() {  
 return "WardRobe{" +  
 "length=" + length +  
 ", width=" + width +  
 ", height=" + height +  
 '}';  
 }  
  
 @Override  
 public void cost(double length,double width,double height) {  
 double cash = (length\*width\*height)\*100;  
 System.out.println("Шафа за заданими розмірами коштує: "+ cash);  
 }  
}

Лістинг 4. Клас Книжкова шафа

package Ex\_2;  
  
public class BookCase extends WardRobe{  
 private String name;  
 private int type;  
 private int numberOfShelves;  
 {  
 name = "Книжкова шафа";  
 type = 1;  
 numberOfShelves = 5;  
 }  
  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public int getType() {  
 return type;  
 }  
 public void setType(int type) {  
 this.type = type;  
 }  
 public int getNumberOfShelves() {  
 return numberOfShelves;  
 }  
 public void setNumberOfShelves(int numberOfShelves) {  
 this.numberOfShelves = numberOfShelves;  
 }  
  
  
 public BookCase(double length, double width, double height, String name, int type, int numberOfShelves) {  
 super(length, width, height);  
 this.name = name;  
 this.type = type;  
 this.numberOfShelves = numberOfShelves;  
 }  
  
 public BookCase(double length, double width, double height) {  
 super(length, width, height);  
 }  
  
 @Override  
 public void cost(double length, double width, double height) {  
 super.cost(length, width, height);  
 }  
 public void cost(double length,double width, double height,int type,int numberOfShelves)  
 {  
 if(type == 1){  
 double cash = ((length\*width\*height\*numberOfShelves)\*100)+150;  
 System.out.println("Шкафа за заданими параметрами коштує:"+cash+" Грн.");  
 }  
 else  
 {  
 double cash = ((length\*width\*height\*numberOfShelves)\*100)+400;  
 System.out.println("Шкафа за заданими параметрами коштує:"+cash+" Грн.");  
 }  
  
 }  
  
 @Override  
 public String toString() {  
 return "BookCase{" +  
 "name='" + name + '\'' +  
 ", type=" + type +  
 ", numberOfShelves=" + numberOfShelves +  
 ", length=" + length +  
 ", width=" + width +  
 ", height=" + height +  
 '}';  
 }  
}

Лістинг 5. Клас Main

package Ex\_2;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class Main {  
 public static void main(String[] args)  
 {  
 Furniture robe = new BookCase(2.4,3.2,4.2,"Шкафа першого типу",1,5);  
 robe.cost(2.4,3.2,4.2,1,5);  
 System.out.println(robe);  
 printTimeStamp();  
 }  
 public static void printTimeStamp() {  
 final String sname = "Yaroslav Solomianyi";  
 Date currentDate = new Date();  
 SimpleDateFormat dateFormat = null;  
 dateFormat = new SimpleDateFormat();  
 System.out.println("Made by: " + sname + " on " + dateFormat.format(currentDate));  
 }  
}

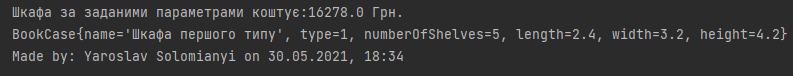


Рис 4. Результат виконання завдання №2.

Завдання 3. Створіть інтерфейс Measurable з методом double getMeasure(), який вимірює об’єкт деяким чином. Клас Employee має реалізовувати Measurable. Забезпечте метод double average(Measurable[] objects), який обчислює усереднену величину вимірів. Використайте метод для обчислення усередненої заробітної плати з масиву робітників. Доповніть інтерфейс Measurable методом largest(Measurable[] objects). Використайте його для знаходження імені робітника з найвищою заробітною платою.

Лістинг 6.Інтерфейс Measurable

package Ex\_3;  
  
public interface Measurable {  
 double getMeasure();  
 String largest();  
}

Лістинг 7.Клас Employee

package Ex\_3;  
  
public class Employee implements Measurable{  
 private String name;  
 private int salary;  
  
 public String getName() {  
 return name;  
 }  
 public int getSalary() {  
 return salary;  
 }  
  
 public Employee(String name, int salary) {  
 this.name = name;  
 this.salary = salary;  
 }  
  
 @Override  
 public double getMeasure()  
 {  
 return salary;  
 }  
  
 @Override  
 public String largest()  
 {  
 return name;  
 }  
}

Лістинг 8. Клас Data

package Ex\_3;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class Data {  
 public static void main(String[] args) {  
 Employee[] SuperEmployee =  
 {  
 new Employee("Yarik",5250),  
 new Employee("Stanislav",5000),  
 new Employee("Maksim",5900)  
 };  
 double temp = average(SuperEmployee);  
 System.out.println(temp);  
 Measurable temp2 = largest(SuperEmployee);  
 System.out.println(temp2.largest());  
 printTimeStamp();  
 }  
 public static double average(Measurable[] objects)  
 {  
 double sum = 0;  
 for (Measurable obj : objects)  
 {  
 sum = sum + obj.getMeasure();  
 }  
 if (objects.length > 0)  
 {  
 double a = Math.round(sum/ objects.length);  
 return a;  
 }  
 else  
 {  
 return 0;  
 }  
 }  
 public static Measurable largest(Measurable[] objects) {  
 Measurable max = null;  
 for (Measurable obj : objects) {  
 if (max == null || obj.getMeasure() > max.getMeasure()) {  
 max = obj;  
 }  
 }  
 return max;  
 }  
 public static void printTimeStamp() {  
 final String sname = "Yaroslav Solomianyi";  
 Date currentDate = new Date();  
 SimpleDateFormat dateFormat = null;  
 dateFormat = new SimpleDateFormat();  
 System.out.println("\nMade by: " + sname + " on " + dateFormat.format(currentDate));  
 }  
}

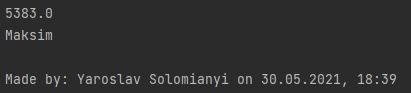


Рис 5. Результат виконання завдання 3.

Завдання 4. Напишіть узагальнений метод selectionSort(), який виконуватиме сортування вибором Integer та Float масивів. У відповідному класі вводьте дані в масив з клавіатури, запускайте сортування та виводьте його результат.

• Підказка: використовуйте > для методу selectionSort(), щоб можна було використовувати метод compareTo() для порівняння об’єктів, представлених типом T.

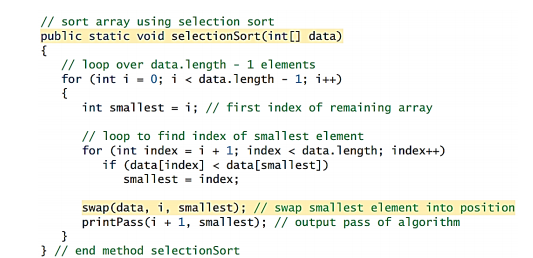
• Псевдокод алгоритму 

Рис 6. Псевдокод алгоритму

Лістинг 9. Виконання завдання 4.

package Ex\_4;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class SelectionSort  
{  
 public static void main(String[] args)  
 {  
 Integer[] list = {28, 77, 43, 11, 55, 9, 1};  
 System.out.println("Оригінальний масив: ");  
 printArray(list);  
  
 selectionSort(list);  
 System.out.println("\nСортування вибіркою:");  
 printArray(list);  
 selectionSort(list);  
 printTimeStamp();  
 }  
 public static <T extends Comparable<T>> void selectionSort(T[] list)  
 {  
 for(int i=0; i<list.length-1; i++)  
 {  
 int iSmallest = i;  
  
 for(int j=i+1; j<list.length; j++)  
 {  
 if(list[iSmallest].compareTo((list[j])) > 0 )  
 {  
 iSmallest = j;  
 }  
 }  
 T iSwap = list[iSmallest];  
 list[iSmallest] = list[i];  
 list[i] = iSwap;  
  
 }  
 }  
  
 public static <T> void printArray(T[] list)  
 {  
  
 for(int i=0; i<list.length; i++)  
 {  
 System.out.print(list[i] + ", ");  
 }  
 }  
 public static void printTimeStamp() {  
 final String sname = "Yaroslav Solomianyi";  
 Date currentDate = new Date();  
 SimpleDateFormat dateFormat = null;  
 dateFormat = new SimpleDateFormat();  
 System.out.println("\nMade by: " + sname + " on " + dateFormat.format(currentDate));  
 }  
}

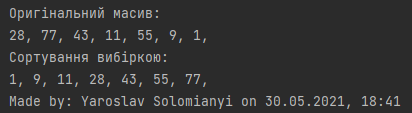


Рис 7. Результат виконання завдання 4.

Завдання 5. Напишіть класи Shape, Circle, Rectangle, Square відповідно до діаграми класів



Рис 8. Задані діаграми класів

Лістинг 10. Абстрактний клас Shape

package ex\_5;  
  
abstract class Shape {  
 protected String color;  
 protected boolean filled;  
  
 public Shape(){}  
 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;  
 }  
 abstract double getArea();  
 abstract double getPerimeter();  
  
 @Override  
 public String toString() {  
 return "Shape{" +  
 "color='" + color + '\'' +  
 ", filled=" + filled +  
 '}';  
 }  
}

Лістинг 11. Клас Circle

package ex\_5;  
  
public class Circle extends Shape{  
 protected double radius;  
  
 public Circle(){}  
  
 public double getRadius() {  
 return radius;  
 }  
  
 public void setRadius(double radius) {  
 this.radius = radius;  
 }  
  
 public Circle(double radius) {  
 this.radius = radius;  
 }  
  
 public Circle(String color, boolean filled, double radius) {  
 super(color, filled);  
 this.radius = radius;  
 }  
 @Override  
 double getArea()  
 {  
 return (Math.PI\*(Math.pow(radius,2)));  
 }  
  
 @Override  
 double getPerimeter()  
 {  
 return (2\*Math.PI\*radius);  
 }  
  
 @Override  
 public String toString() {  
 return "Circle{" +  
 "radius=" + radius +  
 ", color='" + color + '\'' +  
 ", filled=" + filled +  
 '}';  
 }  
}

Лістинг 12. Class Square

package ex\_5;  
  
public class Square extends Rectangle{  
 protected double side;  
 public Square(){}  
 public Square(double side){}  
  
 @Override  
 public String toString() {  
 return "Square{" +  
 "width=" + width +  
 ", length=" + length +  
 ", color='" + color + '\'' +  
 ", filled=" + filled +  
 ", side=" + side +  
 '}';  
 }  
  
 public Square(double width, double length, double side) {  
 super(width, length);  
 this.side = side;  
 }  
  
 public double getSide() {  
 return side;  
 }  
  
 public void setSide(double side) {  
 this.side = side;  
 }  
  
 @Override  
 public void setWidth(double side) {  
 super.setWidth(side);  
 }  
  
 @Override  
 public void setLength(double side) {  
 super.setLength(side);  
 }  
}

Лістинг 13. Class Rectangle

package ex\_5;  
  
public class Rectangle extends Shape{  
 protected double width;  
 protected double length;  
 public Rectangle(){}  
  
 public Rectangle(double width, double length) {  
 this.width = width;  
 this.length = length;  
 }  
  
 public Rectangle(String color, boolean filled, double width, double length) {  
 super(color, filled);  
 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  
 double getArea() {  
 return (length\*width);  
 }  
  
 @Override  
 double getPerimeter() {  
 return ((length+width)\*2);  
 }  
  
 @Override  
 public String toString() {  
 return "Rectangle{" +  
 "width=" + width +  
 ", length=" + length +  
 ", color='" + color + '\'' +  
 ", filled=" + filled +  
 '}';  
 }  
}

Лістинг 14. Class Test

package ex\_5;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class Test {  
 public static void printTimeStamp() {  
 final String sname = "Yaroslav Solomianyi";  
 Date currentDate = new Date();  
 SimpleDateFormat dateFormat = null;  
 dateFormat = new SimpleDateFormat();  
 System.out.println("\nMade by: " + sname + " on " + dateFormat.format(currentDate));  
 }  
 public static void main(String[] args)  
 {  
 printTimeStamp();  
 Shape s1 = new Circle("Red", true, 0.5);//Перший параметр передаємо типу String, Другий Boolean, Третій Double;  
 System.out.println(s1);  
 System.out.println(s1.getArea());  
 System.out.println(s1.getPerimeter());  
 System.out.println(s1.getColor());  
 System.out.println(s1.isFilled());  
 //System.out.println(s1.getRadius()); //Помилка тому-що клас Shape немає методу getRadius();  
  
 Circle c1 = (Circle)s1;  
 System.out.println(c1);  
 System.out.println(c1.getArea());  
 System.out.println(c1.getPerimeter());  
 System.out.println(c1.getColor());  
 System.out.println(c1.isFilled());  
 System.out.println(c1.getRadius());  
  
 //Shape s2 = new Shape();//Немає переданих параметрів  
  
 Shape s3 = new Rectangle("red", true, 0.5, 2.2); // Перший параметр передається типу String,Другий параметр типу булеан, а третій double  
 System.out.println(s3);  
 System.out.println(s3.getArea());  
 System.out.println(s3.getPerimeter());  
 System.out.println(s3.getColor());  
 //System.out.println(s3.getLength());//class Shape не містить методу getLength();  
  
 Rectangle r1 = (Rectangle)s3;  
 System.out.println(r1);  
 System.out.println(r1.getArea());  
 System.out.println(r1.getColor());  
 System.out.println(r1.getLength());  
  
 Shape s4 = new Square(6.6);  
 System.out.println(s4);  
 System.out.println(s4.getArea());  
 System.out.println(s4.getColor());  
 //System.out.println(s4.getSide());//Class Shape не містить методу getSide();  
  
 Rectangle r2 = (Rectangle)s4;  
 System.out.println(r2);  
 System.out.println(r2.getArea());  
 System.out.println(r2.getColor());  
 //System.out.println(r2.getSide());//Class Rectangle не містить методу getSide();  
 System.out.println(r2.getLength());  
  
 Square sq1 = (Square)r2;  
 System.out.println(sq1);  
 System.out.println(sq1.getArea());  
 System.out.println(sq1.getColor());  
 System.out.println(sq1.getSide());  
 System.out.println(sq1.getLength());  
 }  
}

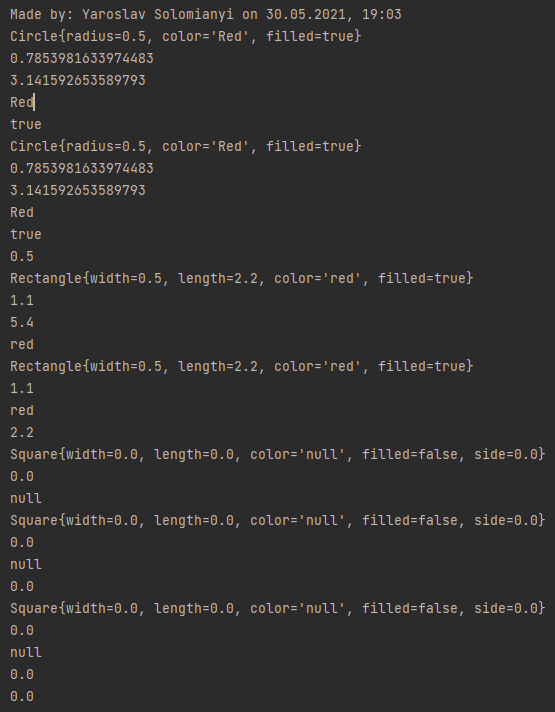


Рис 9.Результат роботи програми 5.

Завдання 6. Припустимо, що Ви маєте набір об'єктів з подібною поведінкою: вони можуть рухатись вгору, вниз, вправо або вліво. Конкретна поведінка (як рухатись та на яку відстань) залежить від самих об’єктів. Поширеним способом моделювання такої поведінки є визначення інтерфейсу Movable з абстрактними методами moveUp(), moveDown(), moveLeft() та moveRight(). Класи, які реалізують інтерфейс Movable, будуть постачати конкретні реалізації абстрактних методів.

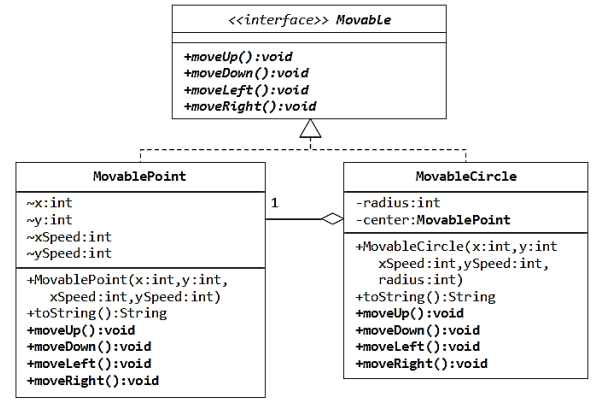


Рис 10. Діаграма класів

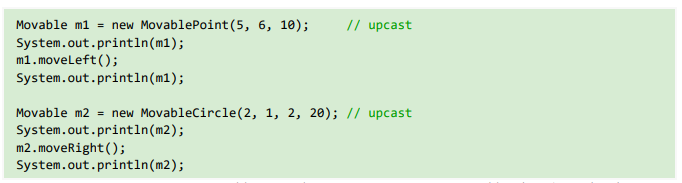
Напишіть два конкретних класи MovablePoint та MovableCircle, які реалізують інтерфейс Movable відповідно до діаграми класів. Реалізуйте тестовий клас та спробуйте додати в нього наступні інструкції: 

Рис 11. Інструкції щодо коду.

Додайте новий клас MovableRectangle, який компонує два MovablePoints (верхній лівий та нижній правий кути) та реалізує інтерфейс Movable, як це показано на діаграмі. Переконайтесь, що обидві точки мають однакову швидкість.

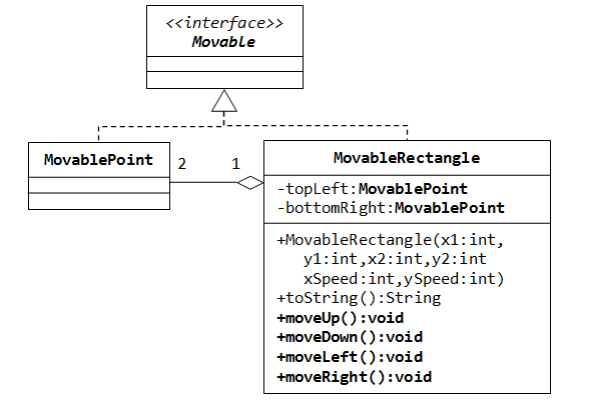


Рис 12.Діаграма класів

Лістинг 15. Інтерфейс Movable

package Ex\_6;  
  
public interface Movable {  
 public void moveUp();  
 public void moveDown();  
 public void moveLeft();  
 public void moveRight();  
}

Лістинг 16. Class MovableCircle

package Ex\_6;  
  
public class MovableCircle implements Movable{  
 private int radius;  
 private MovablePoint center;  
  
 @Override  
 public String toString() {  
 return "MovableCircle{" +  
 "radius=" + radius +  
 ", center=" + center +  
 '}';  
 }  
  
 public MovableCircle(int x, int y, int xSpeed, int ySpeed, int radius) {  
 center = new MovablePoint(x,y,xSpeed,ySpeed);  
 this.radius = radius;  
 }  
  
 @Override  
 public void moveUp() {  
 center.moveUp();  
 }  
  
 @Override  
 public void moveDown() {  
 center.moveDown();  
 }  
  
 @Override  
 public void moveRight() {  
 center.moveRight();  
 }  
  
 @Override  
 public void moveLeft() {  
 center.moveLeft();  
 }  
}

Лістинг 17. Class MovablePoint

package Ex\_6;  
  
public class MovablePoint implements Movable{  
 public MovablePoint(int x, int y, int xSpeed, int ySpeed) {  
 this.x = x;  
 this.y = y;  
 this.xSpeed = xSpeed;  
 this.ySpeed = ySpeed;  
 }  
  
 protected int x;  
 protected int y;  
 protected int xSpeed;  
 protected int ySpeed;  
  
 @Override  
 public void moveUp() {  
 y += ySpeed;  
 }  
  
 @Override  
 public void moveDown() {  
 y -= ySpeed;  
 }  
  
 @Override  
 public void moveLeft() {  
 x -= xSpeed;  
 }  
  
 @Override  
 public void moveRight() {  
 x += xSpeed;  
 }  
  
 @Override  
 public String toString() {  
 return "MovablePoint{" +  
 "x=" + x +  
 ", y=" + y +  
 ", xSpeed=" + xSpeed +  
 ", ySpeed=" + ySpeed +  
 '}';  
 }  
}

Лістинг 18. Class MovableRectangle

package Ex\_6;  
  
public class MovableRectangle implements Movable {  
  
 private MovablePoint topLeft;  
 private MovablePoint bottomRight;  
  
 public MovableRectangle(int x1, int y1, int x2, int y2, int xSpeed, int ySpeed) {  
 topLeft = new MovablePoint(x1, y1, xSpeed, ySpeed);  
 bottomRight = new MovablePoint(x2, y2, xSpeed, ySpeed);  
 }  
  
 @Override  
 public void moveUp() {  
 if ( ! hasPointsSameSpeed()) {  
 return;  
 }  
 topLeft.y -= topLeft.ySpeed;  
 bottomRight.y -= bottomRight.ySpeed;  
 }  
  
 @Override  
 public void moveDown() {  
 if ( ! hasPointsSameSpeed()) {  
 return;  
 }  
 topLeft.y += topLeft.ySpeed;  
 bottomRight.y += bottomRight.ySpeed;  
 }  
  
 @Override  
 public void moveLeft() {  
 if ( ! hasPointsSameSpeed()) {  
 return;  
 }  
 topLeft.x -= topLeft.xSpeed;  
 bottomRight.x -= bottomRight.xSpeed;  
 }  
  
 @Override  
 public void moveRight() {  
 if ( ! hasPointsSameSpeed()) {  
 return;  
 }  
 topLeft.x += topLeft.xSpeed;  
 bottomRight.x += bottomRight.xSpeed;  
 }  
  
 private boolean hasPointsSameSpeed() {  
 return (topLeft.xSpeed == bottomRight.xSpeed)  
 && (topLeft.ySpeed == bottomRight.ySpeed);  
 }  
  
 @Override  
 public String toString() {  
 return String.format("MovableRectangle with topLeft: %1$s and bottomRight: %2$s"  
 , topLeft.toString(), bottomRight.toString());  
 }  
}

Лістинг 19.Class Test

package Ex\_6;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class Test {  
 public static void printTimeStamp() {  
 final String sname = "Yaroslav Solomianyi";  
 Date currentDate = new Date();  
 SimpleDateFormat dateFormat = null;  
 dateFormat = new SimpleDateFormat();  
 System.out.println("\nMade by: " + sname + " on " + dateFormat.format(currentDate));  
 }  
 public static void main(String[] args)  
 {  
 printTimeStamp();  
 Movable m1 = new MovableRectangle(0, 0, 50, 50, 10, 10); // upcast  
 System.out.println(m1);  
 m1.moveLeft();  
 System.out.println(m1);  
  
 Movable m2 = new MovableRectangle(20, 20, 80, 80, 5, 5); // upcast  
 System.out.println(m2);  
 m2.moveRight();  
 System.out.println(m2);  
  
  
 }  
}

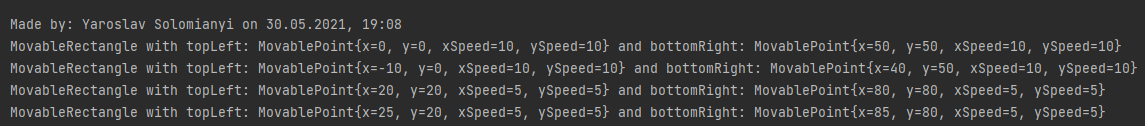


Рис 13.Результат роботи програми 6.