**Interface-Homme Machine / Travaux pratiques**

S01 – Classes, classes abstraites, enum et interfaces





17.03.2015

Adriano De Almeida Silva – T-1f

Alex Travasso – T-1f

**Tables des matières**

[1. Exercices 3](#_Toc414354417)

[1.1. Exercice 1 3](#_Toc414354418)

[1.1.1. Message 3](#_Toc414354419)

[1.1.2. Email 4](#_Toc414354420)

[1.1.3. Sms 5](#_Toc414354421)

[1.1.4. Mms 6](#_Toc414354422)

[1.2. Exercice 2 7](#_Toc414354423)

[1.2.1. Mailbox 7](#_Toc414354424)

[1.3. Exercice 3 8](#_Toc414354425)

[1.3.1. TestMailbox 8](#_Toc414354426)

[1.4. Exercice 4 9](#_Toc414354427)

[1.4.1. Snapchat 9](#_Toc414354428)

[1.4.2. Question 10](#_Toc414354429)

[1.5. Exercice 5 10](#_Toc414354430)

[1.5.1. Rectangle 10](#_Toc414354431)

[1.5.2. RectangleTest 13](#_Toc414354432)

[1.6. Exercice 6 14](#_Toc414354433)

[1.6.1. Box 14](#_Toc414354434)

[1.7. Exercice 7 16](#_Toc414354435)

[1.7.1. Colorable 16](#_Toc414354436)

[1.8. Exercice 8 16](#_Toc414354437)

[1.9. Exercice 9 16](#_Toc414354438)

[1.10. Exercice 10 17](#_Toc414354439)

[1.10.1. ColorTools 17](#_Toc414354440)

[1.10.2. Questions 17](#_Toc414354441)

[1.11. Exercice 11, 12, 13 17](#_Toc414354442)

[Question 19](#_Toc414354443)

# Exercices

## Exercice 1

### Message

package s01;

import java.util.ArrayList;

import javafx.scene.paint.Color;

public abstract class Message implements Colorable {

private String sender;

private ArrayList<String> recipients;

private Color couleur;

public Message(String sender, ArrayList<String> recipients, Color couleur) {

this.sender = sender;

this.recipients = recipients;

this.couleur = couleur;

}

public abstract void display();

public String getSender() {

return sender;

}

public ArrayList<String> getRecipients() {

return recipients;

}

public abstract String getType();

public Color getColor() {

return couleur;

}

public void setColor(Color color) {

this.couleur = color;

}

}

### Email

package s01;

import java.util.ArrayList;

import javafx.scene.paint.Color;

public class Email extends Message {

String message;

public Email(String sender, ArrayList<String> recipients, String message,

Color couleur) {

super(sender, recipients, couleur);

this.message = message;

}

@Override

public void display() {

System.out.println(this);

}

@Override

public String toString() {

return message + " [Couleur:" + super.getColor() + "]";

}

@Override

public String getType() {

return "Email";

}

@Override

public void paint(Color color) {

super.setColor(color);

}

@Override

public Color whatColor() {

return super.getColor();

}

@Override

public void inverseColor() {

if (super.getColor() != null) {

super.setColor(getColor().invert());

}

}

}

### Sms

package s01;

import java.util.ArrayList;

import javafx.scene.paint.Color;

public class Sms extends Message {

char[] messageChar;

public Sms(String sender, ArrayList<String> recipients, String message,

Color couleur) {

super(sender, recipients, couleur);

messageChar = new char[160];

messageToTab(message);

}

private void messageToTab(String message) {

for (int i = 0; i < message.length(); i++) {

messageChar[i] = message.charAt(i);

}

}

@Override

public void display() {

System.out.println(this);

}

@Override

public String toString() {

String message = new String(messageChar);

String color = " [Couleur:" + super.getColor() + "]";

return message+color;

}

@Override

public String getType() {

return "SMS";

}

@Override

public void paint(Color color) {

super.setColor(color);

}

@Override

public Color whatColor() {

return super.getColor();

}

@Override

public void inverseColor() {

if (super.getColor() != null) {

super.setColor(super.getColor().invert());

}

}

}

### Mms

package s01;

import java.util.ArrayList;

import javafx.scene.image.Image;

import javafx.scene.paint.Color;

public class Mms extends Message {

String message;

Image image;

public Mms(String sender, ArrayList<String> recipients, String message,

Image image, Color couleur) {

super(sender, recipients, couleur);

this.message = message;

this.image = image;

}

@Override

public void display() {

System.out.println(this);

}

@Override

public String toString() {

return message + " [Couleur:" + super.getColor() + "]";

}

@Override

public String getType() {

return "MMS";

}

@Override

public void paint(Color color) {

super.setColor(color);

}

@Override

public Color whatColor() {

return super.getColor();

}

@Override

public void inverseColor() {

if (super.getColor() != null) {

super.setColor(super.getColor().invert());

}

}

}

## Exercice 2

### Mailbox

package s01;

import java.util.ArrayList;

import java.util.Random;

import javafx.scene.image.Image;

import javafx.scene.paint.Color;

public class MailBox {

public void send(Message message) {

System.out.println(message.getType() + " envoyé par "

+ message.getSender() + " à "

+ message.getRecipients().toString() + " : "

+ message.toString());

}

public void receive() {

String resultat = "Vous n'avez pas de messages.";

Image image = new Image("http://bit.ly/1zkQXI1");

ArrayList<String> contacts = new ArrayList<String>();

int nbr;

contacts.add("Adriano");

Email mail = new Email("Alex", contacts, "Travail prog", Color.BLACK);

Sms sms = new Sms("Jean", contacts, "Go manger?", Color.YELLOW);

Mms mms = new Mms("Carla", contacts, "Photooo", image, Color.WHITE);

Snapchat snap = new Snapchat("Jean", contacts, image, Color.RED);

Random rdm = new Random();

nbr = rdm.nextInt(100);

if (nbr >= 50 && nbr < 60) {

resultat = "Vous avez reçu un Snapchat de " + snap.getSender()

+ ": " + snap.toString();

} else if (nbr >= 60 && nbr < 80) {

resultat = "Vous avez reçu un E-mail de " + mail.getSender() + ": "

+ mail.toString();

} else if (nbr >= 80 && nbr < 90) {

resultat = "Vous avez reçu un SMS de " + sms.getSender() + ": "

+ sms.toString();

} else if (nbr >= 90 && nbr < 100) {

resultat = "Vous avez reçu un MMS de " + mms.getSender() + ": "

+ mms.toString();

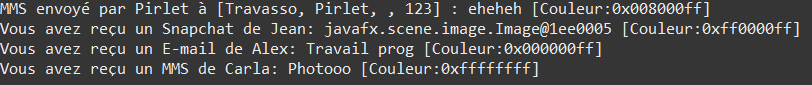
}

System.out.println(resultat);

}

}

Voici un résultat :



## Exercice 3

### TestMailbox

package s01;

import java.util.ArrayList;

import javafx.embed.swing.JFXPanel;

import javafx.scene.image.Image;

import javafx.scene.paint.Color;

public class TestMailbox {

public static void main(String[] args) {

new JFXPanel();

MailBox mailBox = new MailBox();

ArrayList<String> contacts = new ArrayList<String>();

contacts.add("Travasso");

contacts.add("Pirlet");

contacts.add("");

contacts.add("123");

Email mail = new Email("Adriano", contacts, "Salut ça va?", Color.RED);

mailBox.send(mail);

Sms sms = new Sms("Alex", contacts, "Oui et vous?", Color.BLUE);

mailBox.send(sms);

Image image = new Image("http://bit.ly/1zkQXI1");

Mms mms = new Mms("Pirlet", contacts, "eheheh", image, Color.GREEN);

mailBox.send(mms);

mailBox.receive();

mailBox.receive();

mailBox.receive();

}

}

## Exercice 4

### Snapchat

package s01;

import java.util.ArrayList;

import javafx.scene.image.Image;

import javafx.scene.paint.Color;

public class Snapchat extends Message{

Image image;

public Snapchat(String sender, ArrayList<String> recipients,Image image, Color couleur) {

super(sender, recipients, couleur);

this.image = image;

}

@Override

public void paint(Color color) {

// TODO Auto-generated method stub

}

@Override

public Color whatColor() {

return super.getColor();

}

@Override

public void inverseColor() {

if (super.getColor() != null) {

super.setColor(super.getColor().invert());

}

}

@Override

public void display() {

System.out.println(this);

}

@Override

public String getType() {

return "SnapChat";

}

@Override

public String toString() {

return image.toString() + " [Couleur:" + super.getColor() + "]";

}

}

### Question

*Quelles adaptions ont-elles été nécessaires dans la classe MailBox et dans le programme de test ?*

Dans la classe MailBox nous avons dû adapter le système de gestion aléatoire de messages, donc ajouter avec une certaine probabilité d’apparition la classe SnapChat. Aucune adaptation particulière n’as été nécessaire pour la classe de test.

## Exercice 5

### Rectangle

package s01;

import javafx.geometry.Point2D;

public class Rectangle {

private double x0, y0, x1, y1;

private Point2D downLeft, upRight;

Rectangle(double x0, double y0, double x1, double y1) {

this.x0 = x0;

this.y0 = y0;

this.x1 = x1;

this.y1 = y1;

}

Rectangle(Point2D downLeft, Point2D upRight) {

this.downLeft = downLeft;

this.upRight = upRight;

x0 = downLeft.getX();

y0 = downLeft.getY();

x1 = upRight.getX();

y1 = upRight.getY();

}

public double getWidth() {

return x1 - x0;

}

public double getHeight() {

return y1 - y0;

}

public boolean isIn(Point2D point) {

double x = point.getX();

double y = point.getY();

boolean xOkay = x0 < x && x < x1;

boolean yOkay = y0 < y && y < y1;

return xOkay && yOkay;

}

public void move(double x, double y) {

x0 += x;

x1 += x;

y0 += y;

y1 += y;

}

public Rectangle union(Rectangle rec) {

Rectangle resultat = new Rectangle(rec.x0, rec.x1, rec.y0, rec.y1);

if (x0 < rec.x0) {

resultat.x0 = x0;

}

if (x1 > rec.x1) {

resultat.x1 = x1;

}

if (y0 < rec.y0) {

resultat.y0 = y0;

}

if (y1 > rec.y1) {

resultat.y1 = y1;

}

return resultat;

}

public Rectangle intersection(Rectangle rec) {

Rectangle resultat = new Rectangle(rec.x0, rec.y0, rec.x1, rec.y1);

if (x0 > rec.x0 && x0 < rec.x1) {

resultat.x0 = x0;

}

if (x1 < rec.x1 && x1 > rec.x0) {

resultat.x1 = x1;

}

if (y0 > rec.y0 && y0 < rec.y1) {

resultat.y0 = y0;

}

if (y1 < rec.y1 && y1 > rec.y0) {

resultat.y1 = y1;

}

return resultat;

}

@Override

public String toString() {

return "Rectangle [" + x0 + ", " + y0 + "], [" + x1 + ", " + y1 + "]";

}

@Override

public boolean equals(Object ob) {

Rectangle rec = (Rectangle) ob;

if (rec.x0 == x0 && rec.y1 == y1) {

return true;

} else {

return false;

}

}

public double getX0() {

return x0;

}

public void setX0(double x0) {

this.x0 = x0;

}

public double getY0() {

return y0;

}

public void setY0(double y0) {

this.y0 = y0;

}

public double getX1() {

return x1;

}

public void setX1(double x1) {

this.x1 = x1;

}

public double getY1() {

return y1;

}

public void setY1(double y1) {

this.y1 = y1;

} }

### RectangleTest

package s01;

import javafx.geometry.Point2D;

import javafx.scene.paint.Color;

public class RectangleTest {

public static void main(String[] args) {

Rectangle rec1 = new Rectangle(4, 6, 8, 8);

Rectangle rec2 = new Rectangle(7, 7, 10, 9);

Rectangle rec3 = new Rectangle(4, 6, 8, 8);

Point2D point = new Point2D(5, 7);

Rectangle intersec = rec1.intersection(rec2);

Rectangle uni = rec1.union(rec2);

Box box1 = new Box(0, 0, 2, 2, 1.0, LineType.DASHED, Color.RED);

Box box2 = new Box(0, 0, 2, 2, 1.0, LineType.DASHED, Color.BLUE);

Box box3 = new Box(1, 1, 3, 3, 2.0, LineType.PLAIN, Color.GREEN);

boolean resultat = true;

if (!rec1.equals(rec3))

resultat = false;

if (rec2.getHeight() != 2)

resultat = false;

if (rec2.getWidth() != 3)

resultat = false;

if (!rec1.isIn(point))

resultat = false;

rec2.move(1, 1);

if (rec2.getX0() != 8 || rec2.getY0() != 8)

resultat = false;

// rec2 (8, 8, 11, 10)

rec2.move(-1, -1);

if (uni.getX0() != 4 || uni.getY1() != 9)

resultat = false;

if (intersec.getX0() != 7 || intersec.getY1() != 8)

resultat = false;

System.out.println(intersec.toString());

if (!box1.equals(box2))

resultat = false;

if (box1.equals(box3))

resultat = false;

System.out.println(box1.toString());

if (resultat == true) {

System.out.println("Test passed successfully");

} else {

System.out.println("Something is wrong...");

}

}

}

## Exercice 6

### Box

package s01;

import javafx.geometry.Point2D;

import javafx.scene.paint.Color;

public class Box extends Rectangle implements Colorable {

private double epaisseur;

private LineType typeLigne;

private Color couleur;

public Box(double x0, double y0, double x1, double y1, double epaisseur,

LineType typeLigne, Color couleur) {

super(x0, y0, x1, y1);

this.epaisseur = epaisseur;

this.typeLigne = typeLigne;

this.couleur = couleur;

}

public Box(Point2D downLeft, Point2D upRight, double epaisseur,

LineType typeLigne, Color couleur) {

super(downLeft, upRight);

this.epaisseur = epaisseur;

this.typeLigne = typeLigne;

this.couleur = couleur;

}

public double getEpaisseur() {

return epaisseur;

}

public void setEpaisseur(double epaisseur) {

this.epaisseur = epaisseur;

}

public LineType getTypeLigne() {

return typeLigne;

}

public void setTypeLigne(LineType typeLigne) {

this.typeLigne = typeLigne;

}

public double getX0() {

return super.getX0();

}

public double getX1() {

return super.getX1();

}

public double getY0() {

return super.getY0();

}

public double getY1() {

return super.getY1();

}

@Override

public String toString() {

return "Box [" + getX0() + ", " + getY0() + "], [" + getX1() + ", "

+ getY1() + "]" + " [Epaisseur:" + epaisseur + "] " + "[type:"

+ typeLigne.toString() + "]" + " [Couleur:" + couleur + "]";

}

@Override

public boolean equals(Object ob) {

Box box = (Box) ob;

if (getX0() == box.getX0() && getY1() == box.getY1()

&& epaisseur == box.getEpaisseur()

&& typeLigne == box.getTypeLigne()) {

return true;

} else {

return false;

}

}

@Override

public void paint(Color color) {

this.couleur = color;

}

@Override

public Color whatColor() {

return couleur;

}

@Override

public void inverseColor() {

if (couleur != null) {

couleur.invert();

}

}

}

## Exercice 7

### Colorable

package s01;

import javafx.scene.paint.Color;

public interface Colorable {

// permet de définir la couleur de l'objet

public void paint(Color color);

// permet de connaitre sa couleur(null si pas definie)

public Color whatColor();

//inverse la couleur actuellement définie (sans effet si aucune couleur n'est definie

public void inverseColor();

}

## Exercice 8

La classe Box est décrite à l’exercice 6 avec l’implémentation de l’interface colorable.

## Exercice 9

La classe Message est décrite à l’exercice 1 avec l’implémentation de l’interface colorable.

## Exercice 10

### ColorTools

package s01;

import javafx.scene.paint.Color;

public class ColorTools {

public static void paintALL(Colorable[] tabColor, Color couleur) {

for (Colorable colorable : tabColor) {

colorable.paint(couleur);

}

}

}

### Questions

*De quel type faut-il définir ce tableau ?*

Il faut définir le tableau de type Colorable.

*Déclarer le tableau de type Object[] est certainement possible mais quelles en sont les implications ?*

Les implications sont qu’il faut caster les objets et tester toutes les erreurs possibles, car l’objet à caster ne peut pas être n’importe lequel.

## Exercice 11, 12, 13

**package** s01File;

**import** java.io.File;

**import** java.io.FilenameFilter;

**public** **class** TestFiles {

**public** **static** **void** main(String[] args) {

// 11

// fichier existant

File e = **new** File("test.txt");

// fichier inéxistant

File i = **new** File("inconnu.txt");

// répertoire

File d = **new** File("src");

// test fichier éxistant

System.***out***.println("Chemin absolu : " + e.getAbsolutePath());

System.***out***.println("Nom du fichier : " + e.getName());

System.***out***.println("Est-ce qu'il existe ? " + e.exists());

System.***out***.println("Est-ce un répertoire ? " + e.isDirectory());

System.***out***.println("Est-ce un fichier ? " + e.isFile());

System.***out***.println("Taille ? " + e.getTotalSpace() + " bytes");

System.***out***.println();

// test fichier inéxistant

System.***out***.println("Chemin absolu : " + i.getAbsolutePath());

System.***out***.println("Nom du fichier : " + i.getName());

System.***out***.println("Est-ce qu'il existe ? " + i.exists());

System.***out***.println("Est-ce un répertoire ? " + i.isDirectory());

System.***out***.println("Est-ce un fichier ? " + i.isFile());

System.***out***.println("Taille ? " + i.getTotalSpace() + " bytes");

System.***out***.println();

// test répertoire

System.***out***.println("Chemin absolu : " + d.getAbsolutePath());

System.***out***.println("Nom du fichier : " + d.getName());

System.***out***.println("Est-ce qu'il existe ? " + d.exists());

System.***out***.println("Est-ce un répertoire ? " + d.isDirectory());

System.***out***.println("Est-ce un fichier ? " + d.isFile());

System.***out***.println("Taille ? " + d.getTotalSpace() + " bytes");

System.***out***.println();

// 12

System.***out***.println("Contenu du répertoire " + d);

String[] paths;

**try** {

// on utilise le répertoire d défini au début de la classe

paths = d.list();

**for** (String path : paths) {

System.***out***.println(path);

}

} **catch** (Exception t) {

t.printStackTrace();

}

System.***out***.println();

// 13

// Classe anonyme avec un filtre

String[] fileListFilter = d.list(**new** FilenameFilter() {

**public** **boolean** accept(File dir, String s) {

**return** s.endsWith(".txt");

}

});

// lambda

String[] fileListLambda = d.list((File dir, String s) -> {

**return** s.endsWith(".jpg");

});

System.***out***.println("Répertoire trié avec '. txt' : ");

**for** (**int** t = 0; t < fileListFilter.length; t++) {

System.***out***.println(fileListFilter[t]);

}

System.***out***.println("Répertoire trié avec '. jpg' : ");

**for** (**int** r = 0; r < fileListLambda.length; r++) {

System.***out***.println(fileListLambda[r]);

}

}

}

### Question

A votre avis, quelle est la meilleure variante ?

*La méthode lambda, car c’est la plus simple à mettre en place*