Daouda KONE

Exercice 1

Question 1 et 2

public record Book(String title, String author){

```
public static void main(String[] args){
    var book = new Book("Da Vinci Code", "Dan Brown");
    System.out.println(book.title() + ' ' + book.author());

}
```

}

Explication: Ici nous sommes dans un *record* public mais ses attributs **title** et **author** sont privés. Le **main** marche parce que nous sommes dans le même fichier. . . ..

Question 3

public class Main{ public static void main(String[] args){ var book = new Book(“Da Vinci Code”, “Dan Brown”); System.out.println(book.title() + ’ ’ + book.author());

```
}
```

}

Lorsque nous déplaçons la méthode main dans une classe Main, nous devons tenir compte du fait que les champs des records ne sont pas accessibles comme des champs publics. Pour allons utiliser les methodes d’accès **title()** et **author()** générées automatoquement par *record*.

Question 4

import java.util.Objects;

public record Book(String title, String author) { public Book(String title, String author) { this.title = Objects.requireNonNull(title, “Le titre ne peut pas être null”); this.author = Objects.requireNonNull(author, “L’auteur ne peut pas être null”); } }

public class Main{ public static void main(String[] args){ var book = new Book(“Da Vinci Code”, “Dan Brown”); var weirdBook = new Book(null, “oops”); System.out.println(weirdBook.title() + ’ ’ + weirdBook.author());

```
}
```

}

Et j'obtiens dans quelque choose dans mon terminale:

kddaouda@Daouda:~/Esiee/Java/tp3$ java Main Exception in thread "main" java.lang.NullPointerException: Le titre ne peut pas être null at java.base/java.util.Objects.requireNonNull(Objects.java:246) at Book.(Book.java:6) at Main.main(Main.java:4)

Question 5

Dans mon cas, j'ai opter pour la méthode compacte pour ne ré-écrire à chaque fois la même chose si c'est demandé dans la suite du tp.

import java.util.Objects;

public record Book(String title, String author){

```
public Book {
    Objects.requireNonNull(title, "The title can't Null");
    Objects.requireNonNull(author, "the author can't Null");
}
```

}

public class Main{ public static void main(String[] args){ var book = new Book("Da Vinci Code", "Dan Brown"); var weirdBook = new Book(null, "oops"); System.out.println(weirdBook.title() + ' ' + weirdBook.author());

```
}
```

}

Et j'obtiens dans quelque choose dans mon terminale:

kddaouda@Daouda:~/Esiee/Java/tp3$ java Main Exception in thread "main" java.lang.NullPointerException: The title can't Null at java.base/java.util.Objects.requireNonNull(Objects.java:2
at Book.(Book.java:6) at Main.main(Main.java:4)

Question 6

import java.util.Objects;

public record Book(String title, String author){

```
public Book {
    Objects.requireNonNull(title, "The title can't Null");
    Objects.requireNonNull(author, "the author can't Null");
}

public Book(String title){
    this(title, "No author");
```

}

}

public class Main{ public static void main(String[] args){ var book = new Book(“Da Vinci Code”, “Dan Brown”); //var weirdBook = new Book(null, “oops”); var book2 = new Book(“daouda”); System.out.println(book2);

}

}

Question 7

On voit que tous les constructeurs ont des arguments **String** donc le compilateur va chercher le constructeur qui a un seul argument *String*.

Question 8

Le code suivant ne fonctionne pas parce qu'un record en Java est immuable.

On peut crée un record qui renvoie un nouveau tittre car on ne peut pas modifier le champs du record.

public Book withTitle(String newTitle) { return new Book(newTitle, this.author); }

public class Main{ public static void main(String[] args){ var book = new Book(“Da Vinci Code”, “Dan Brown”); //var weirdBook = new Book(null, “oops”); var book2 = new Book(“daouda”); var book3 = book2.withTitle(“uzfh”); System.out.println(book3);

}

}

Exercice 2

Question 1

System.out.println(b1 == b2); va retourner *true* parce que b2 est une référence à b1. En Java, l'opérateur == compare les références des objets, pas leur contenu. Comme b2 pointe vers le même objet que b1, la comparaison retourne *true*. Ainsi donc, **System.out.println(b1 == b3);** va retourner *false* car b1 et b3 sont des objects différents.

Question 2

Pour comparer les contenus de deux objects, on utiliser la méthode **equals()** qui est normalement générée automatiquement par *record*.

public class Test{ public static void main(String[] args){ var b1 = new Book(“Da Java Code”, “Duke Brown”); var b2 = b1; var b3 = new Book(“Da Java Code”, “Duke Brown”);

```
    System.out.println(b1.equals(b2));
    System.out.println(b1.equals(b3));
}
```

}

Dans le terminale on a : kddaouda@Daouda:~/Esiee/Java/tp3$ javac Book.java Test.java kddaouda@Daouda:~/Esiee/Java/tp3$ java Test true true

Question 3

j’ai mis ce bout de code dans Book.java:

public boolean isFromTheSameAuthor(Book other){ return author.equals(other.author);

}

Pour l’affichage: System.out.println(book1.isFromTheSameAuthor(book2));

Questions 4 et 5

Pour avoir ce affichage, on va modifier **toString**.

@Override public String toString(){ return title +” By ” + author ; }

Question 6

L’annotation **@Override** demande au compilateur de v´erifier que la m´ethode que l’on veut remplacer existe bien. Elle aide à s’assurer que vous redéfinissez bien la méthode prévue et prévient les erreurs liées à une mauvaise signature.

Exercice 3

question 1

Ici le problème le problème se trouve au niveau de **equals** car nous ne sommes pas dans un *record* mais dans une *classe*. L’implémentation par défaut de equals vérifie si les deux objets sont exactement les mêmes en mémoire, c’est-à-dire si elles sont identiques (si elles pointent vers la même référence) parce que nous sommes dans une *classe*.

Question 2

Pour corriger le problème si on veut forcement utiliser une classe, il faut redéfinir la méthode **equals** dans la classe.

public class Book2 { private final String title; private final String author;

public Book2(String title, String author) { this.title = title; this.author = author; }

@Override public boolean equals(Object obj) { return obj instanceof Book2 book && title.equals(book.title) && author.equals(book.author);

}

@Override public int hashCode() { return Objects.hash(title, author); }

```
public static void main(String[] args) {
  var book1 = new Book2("Da Vinci Code", "Dan Brown");
  var book2 = new Book2("Da Vinci Code", "Dan Brown");
  System.out.println(book1.equals(book2));
}
```

}

Exercice 4

Le fichier Price.java

import java.util.Objects;

public record Price(int gold, int silver) { public static final int SILVER_PER_GOLD = 13;

```
public Price(int gold, int silver) {

    if(gold < 0 || silver < 0){
        throw new IllegalArgumentException("Gold and silver cannot be negative");

    }

    this.gold = gold + silver / SILVER_PER_GOLD;
    this.silver = silver % SILVER_PER_GOLD;
}

public boolean canAfford(Price other) {
    Objects.requireNonNull(other);
    return toSilver() >= other.toSilver();
}

public Price subtract(Price other) {
    Objects.requireNonNull(other);
    int totalSilver = toSilver() - other.toSilver();
    return fromSilver(totalSilver);
```

```
}

public static Price fromSilver(int totalSilver) {
    Objects.requireNonNull(totalSilver);
    return new Price(totalSilver / SILVER_PER_GOLD, totalSilver % SILVER_PER_GOLD);
}

public int toSilver() {
    return gold * SILVER_PER_GOLD + silver;
}

@Override
public String toString() {
    return gold + "g and " + silver + "s";
}
```

}

Le fichier Rogue.java

import java.util.Scanner; import java.util.concurrent.ThreadLocalRandom;

public class Rogue { public static String ask(Scanner scanner) { while (true) { System.out.println(“Type yes (to buy), no (to skip) or quit (to leave)”); var choice = scanner.nextLine(); if (choice.equals(“yes”) || choice.equals(“no”) || choice.equals(“quit”)) { return choice; } System.out.println(“Invalid choice, please try again.”); } }

```
public static void main(String[] args) {
    var rng = ThreadLocalRandom.current();
    var purse = new Price(rng.nextInt(100), rng.nextInt(13));
    var scanner = new Scanner(System.in);
    var potions = 0;

    while (true) {
        System.out.println("You have " + purse + " gold pieces.");
        var price = new Price(rng.nextInt(10), rng.nextInt(13));
        System.out.println("Do you want to buy a potion of healing for " + price + "?");
        var choice = ask(scanner);

        if (choice.equals("quit")) {
            break;
        }
        if (choice.equals("no")) {
            continue;
        }
```

```
        if (!purse.canAfford(price)) {
            System.out.println("You don't have enough gold pieces.");
            continue;
        }

        potions++;
        purse = purse.subtract(price);
    }
    System.out.println("You manage to acquire " + potions + " potions with " + purse + " rem
}
```

}

Daouda KONE