Faculté des Sciences
Aix*Marseille Université

Année universitaire

Site : ☐ Luminy ☐ St-Charles ☐ St-Jérôme ☐ Cht-Gombert ☐ Aix-Montperrin ☐ Aubagne-SATIS
Sujet de : ☐ 1er semestre ☐ 2ème semestre ☒ Session 2 Durée de l'épreuve : 4h
Examen de : L2 Nom du diplôme : Licence d'informatique
Code du module : SIN3U02 Libellé du module : Programmation 2
Calculatrices autorisées : NON Documents autorisés : OUI

---

1 Consignes

Inscrivez votre nom et prénom ci-dessous :
Nom :

Prénom :

Répondez directement sur le sujet en cochant ou en écrivant vos réponses aux emplacements prévus à cet effet.

2 Questions de cours

2.1 Question sur les exceptions

On considère le code suivant :

```
static int get(int[] integers, int index){
    try {
        return integers[index];
    }
    catch (NullPointerException e){
        return 0;
    }
    catch (ArrayIndexOutOfBoundsException e){
        return -1;
    }
}

public static void main(String[] args){
    int[] integers = new int[2];
    integers[0] = 1;
    integers[1] = 2;
    for(int index = 0; index < 4; index++){
        System.out.print(get(null, index) + " ");
        System.out.print(get(integers, index) + " ");
    }
}
```

Quel est l'affichage qui est produit par l'exécution de la méthode `main` ce code ? Cochez la bonne réponse.
☐ 0 0 0 0 0 0 0 0
☐ -1 1 -1 2 -1 0 -1 0
☐ 0 1 0 2 0 0 0 0
☐ 0 1 0 2 0 -1 0 -1
☐ pas d'affichage car le code ne compile pas
☐ pas d'affichage car l'exécution de ce code lève une exception

2.2 Question sur les types

Pour chaque ligne du tableau ci-dessous cochez le ou les qualificatifs qui sont vrais pour le type de la colonne de gauche :

type	classe	interface	type primitif	type paramétré
`List<T>`	☐	☐	☐	☐
`int`	☐	☐	☐	☐
`ArrayList<T>`	☐	☐	☐	☐
`Integer`	☐	☐	☐	☐

2.3 Question sur les définitions

Que peut-on définir à l'intérieur de la déclaration d'une classe abstraite, d'une classe concrète (classe n'étant pas abstraite), d'une interface ou d'une énumération (`enum`) ? Pour chaque ligne du tableau ci-dessous cochez chaque élément qui peut être défini à l'intérieur de la déclaration d'un type de la colonne de gauche :

type	attribut d'instance	constructeur	code de méthode	signature de méthode
classe concrète	☐	☐	☐	☐
classe abstraite	☐	☐	☐	☐
interface	☐	☐	☐	☐
enum	☐	☐	☐	☐

2.4 Question sur l'égalité

On considère deux listes (`list1` et `list2`) instanciées puis modifiées par le code suivant :

```
List<Integer> list1 = new ArrayList<>();
List<Integer> list2 = new ArrayList<>();

list1.add(42);
list2.add(42);
```

Quelle est la valeur de `list1 == list2` ? Répondez directement en dessous.

Quelle est la valeur de `list1.equals(list2)` ? Répondez directement en dessous.

2.5 Question sur les types paramétrés

Dans une classe `MyClass<T>`, quel type d'attribut est valide en Java ? On suppose qu'on a importé `Map` et `List` dans la classe. Cochez la ou les bonnes réponses.

☐ `List<List<T>>`
☐ `List<T>[]`
☐ `T<List<T>>`
☐ `List<T[]>`
☐ `Map<T,T>`
☐ `T[][]`

2.6 Question sur les tableaux

On considère le code suivant :

```
String[][] matrixOfStrings = new String[3][];

for(int row = 0; row < matrixOfStrings.length; row++) {
    matrixOfStrings[row] = new String[row];
    for (int column = 0; column < matrixOfStrings[row].length; column++)
        matrixOfStrings[row][column] = "(" + row + ", " + column + ")";
}

for(String[] line : matrixOfStrings)
    for(String value : line)
        System.out.print(value);
```

Quel est l'affichage qui est produit par ce code ? Cochez la bonne réponse.
□ (0, 0)(1, 0)(1, 1)(2, 0)(2, 1)(2, 2)
□ (0, 0)(0, 1)(0, 2)(1, 0)(1, 1)(1, 2)(2, 0)(2, 1)(2, 2)
□ (1, 0)(2, 0)(2, 1)
□ (1, 1)(1, 2)(1, 3)(2, 1)(2, 2)(2, 3)(3, 1)(3, 2)(3, 3)
□ pas d'affichage car le code ne compile pas
□ pas d'affichage car l'exécution de ce code lève une exception

3 Extension de classe

On considère deux classes `Point` et `Pixel` définies par le code suivant :

```
public class Point {
    final int x;
    final int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}

public class Pixel extends Point {
    Color color;

    public Pixel(int x, int y, Color color) {
        /* code manquant ? */
        this.color = color;
    }
}
```

3.1 Question code manquant

Quelle est la partie de code qui manque dans la première ligne du constructeur de `Pixel` (à la place du commentaire) ? Cochez la bonne réponse.
□ `this()`
□ `this(x,y)`
□ `super()`
□ `super(x,y)`
□ Il ne manque pas de code !

3.2 Question sur les types (1/2)

On considère trois listes déclarées et initialisées par le code suivant :

```
List<Pixel> pixels = new ArrayList<>();
List<Point> points = new ArrayList<>();
List<Object> objects = new ArrayList<>();
```

En sachant que la méthode `add` de `List<E>` a pour signature `boolean add(E e)`. Quels sont les appels de `add` qui ne produisent pas d'erreurs à la compilation ? Cochez la ou les bonnes réponses.

☐ `pixels.add(new Point(0, 0))`
☐ `pixels.add(new Object())`
☐ `points.add(new Point(1, 1))`
☐ `points.add(new Pixel(0, 0, Color.BLACK))`
☐ `points.add(new Object())`
☐ `objects.add(new Object())`
☐ `objects.add(new Point(2, 2))`
☐ `objects.add(new Pixel(0, 0, Color.BLACK)`

3.3 Question sur les types (2/2)

Quels sont les affectations qui ne produisent pas d'erreurs à la compilation ? Cochez la ou les bonnes réponses.

☐ `Point point = new Point(1, 1)`
☐ `Point point = new Pixel(0, 0, Color.BLACK)`
☐ `Point point = new Object()`
☐ `Pixel pixel = new Point(1, 1)`
☐ `Pixel pixel = new Object()`
☐ `Object object = new Point(0, 0)`
☐ `Object object = new Pixel(0, 0, Color.BLACK)`

4 Gestion d'un hôpital

On souhaite gérer un hôpital en utilisant les classes suivantes :

- `People` représente une personne.
- `Employee` représente les personnes travaillant à l'hôpital.
- `Surgeon` représente les chirurgiens de l'hôpital. Les salaires des chirurgiens sont calculés en fonction du nombre d'opérations et du type d'opérations qu'ils effectuent.
- `Nurse` représente les infirmiers de l'hôpital. Les salaires des infirmiers sont calculés en fonction du nombre de gardes qu'ils effectuent.
- `Patient` représente les patients de l'hôpital. Chaque patient a une date de sortie prévue de l'hôpital.
- `Date` représente une date.
- `PersonalInformation` représente les coordonnées d'une personne : nom, prénom, date de naissance, adresse, numéro de téléphone, courriel.
- `SurgicalOperation` représente une opération chirurgicale sur un patient ayant un coût, une date, des chirurgiens et infirmiers impliqués et une spécialité.
- `SurgicalSpecialty` représente une spécialité comme *cardiac* (cardiovasculaire), *general* (digestive), *orthopedic* (orthopédie) ou *plastic*.

Je vous conseille de bien réfléchir aux réponses aux questions suivantes. La cohérence des réponses sera évaluée. Cela signifie que vous pouvez perdre des points si vous avez fait des choix dans les premières questions qui sont incompatibles avec vos choix dans les questions suivantes.

4.1 Question sur les types

Pour chaque ligne du tableau ci-dessous cochez le ou les qualificatifs qui sont vrais selon vous pour le type dans la colonne de gauche :

type	classe concrète	interface	classe abstraite	enum
`People`	☐	☐	☐	☐
`Employee`	☐	☐	☐	☐
`Surgeon`	☐	☐	☐	☐
`Nurse`	☐	☐	☐	☐
`Patient`	☐	☐	☐	☐
`Date`	☐	☐	☐	☐
`PersonalInformation`	☐	☐	☐	☐
`SurgicalOperation`	☐	☐	☐	☐
`SurgicalSpecialty`	☐	☐	☐	☐

4.2 Question sur la déclaration de méthode

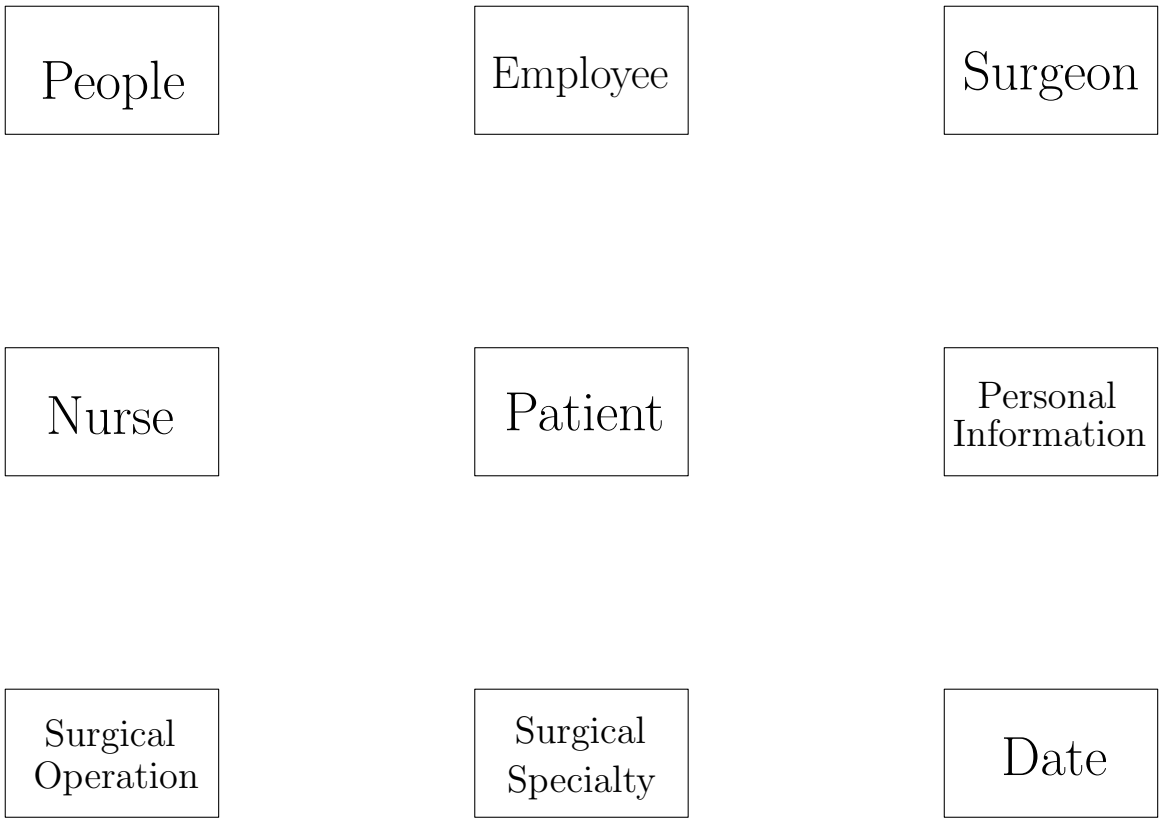
Sur quel type d'objet peut-on selon vous appeler la méthode `getSalary` (*salary* = salaire en anglais) ? Cochez la ou les bonnes réponses.
☐ `People`
☐ `Employee`
☐ `Surgeon`
☐ `Nurse`
☐ `Patient`
☐ `Date`
☐ `PersonalInformation`
☐ `SurgicalOperation`
☐ `SurgicalSpecialty`

4.3 Question sur la déclaration d'attribut

Quel type d'objet possède selon vous un attribut nommé `emailAddress` ? Cochez la ou les bonnes réponses.
☐ `People`
☐ `Employee`
☐ `Surgeon`
☐ `Nurse`
☐ `Patient`
☐ `Date`
☐ `PersonalInformation`
☐ `SurgicalOperation`
☐ `SurgicalSpecialty`

4.4 Question sur la composition/agrégation

Connecter d'une flèche les types qui ont des instances contenues dans un autre type dans la figure en haut de la page suivante. Plus précisément, tracer une flèche allant du type `A` au type `B` si `B` est une classe contenant un attribut de type `A` ou une collection de type `A` (de type `Collection<A>`).



4.5 Question sur l'héritage

Connecter d'une flèche les types qui hérite d'autre type. Plus précisément, tracer une flèche de trait continu allant du type `A` au type `B` si le type `A` étend (mot-clé `extends`) le type `B` (`A` → `B`) et tracer une flèche en pointillés allant du type `A` au type `B` si le type `A` implémente (mot-clés `implements`) le type `B` (`A` ⇢ `B`).