Ciências ULisboa | Informática

Programação Centrada em Objetos

2021/2022

Série 6

Exercícios

1. No exercício 1 da Série 5, a classe abstrata `Contest` tem dois métodos abstratos:

- `boolean isEligible(String cand)` que devolve `true` se o candidato `cand` é elegível para este concurso;
- `int contestEvaluation(String cand)` que devolve a avaliação dada ao candidato `cand` neste concurso (que depende, de alguma forma, das avaliações desse candidato a cada critério de avaliação).

Suponha antes que nesta classe `Contest` queremos já implementar o método `isEligible` de uma forma que consideramos ser suficientemente geral para se aplicar a alguns possíveis tipos de concurso:

- `boolean isEligible(String cand)` que devolve `true` se as notas que o candidato `cand` tem nos vários critérios estão no intervalo [`minGrade()`, `maxGrade()`];

Suponha ainda que na subclasse `MyKindOfContest` o conceito de elegibilidade é dado por:

- um candidato é elegível se as notas que o candidato `cand` tem nos vários critérios estão no intervalo [`minGrade()`, `maxGrade()`] e a média das notas que tem nos critérios `ACADEMIC_GRADE`, `PROFESSIONAL_EXPERIENCE` e `COURSE_RANKING` é igual ou superior a `(maxGrade()- 1);`

a) Altere as classes `Contest` e `MyKindOfContest` de modo a refletirem estes novos conceitos.

b) Reflita sobre o contrato do método `contestEvaluation` nas várias classes envolvidas.

2. Considere as classes que se seguem. Modifique estas classes de forma a terem uma noção de igualdade apropriada.

```
public class Rectangle {
        // The width and height of the rectangle
        // @invariant width > 0 && height > 0
        private int width;
        private int height;

        /**
         * Constructor of a rectangle with dimensions w x h
         * @param w the width
         * @param h the height
         * @requires w > 0 && h > 0
         * @ensures getWidth() == w && getHeight() == h;
         */
        public Rectangle(int w, int h){
```

```
                width = w;
                height = h;
        }

        public double perimeter(){
                return 2 * (width + height);
        }

        /**
         * Resizes the rectangle, maintaining the ratio
         * @param perc the percentage
         * @requires perc > 0;
         * @ensures getWidth() == \old(getWidth()) * perc / 100 &&
         *          getHeight() == \old(getHeight()) * perc / 100;
         */
        public void resize(int perc) {
                width = (width * perc) / 100;
                height = (height * perc) / 100;
        }

        public int getWidth() { return width; }

        public int getHeight() { return height; }
}



public class Color {

    private final static int CTE = 2;

    // @invariant 0 <= r, g, b <= 255
    private int red;
    private int green;
    private int blue;

    // @requires 0 <= r, g, b <= 255
    public Color(int r, int g,int b) {
        red = r;
        green  = g;
        blue = b;
    }

    public void darken() {
      red = red - CTE >= 0 ? red - CTE : 0;
      blue = blue - CTE >= 0 ? blue - CTE : 0;
      green = green - CTE >= 0 ? green - CTE : 0;
    }
}



public class ColoredRectangle extends Rectangle {

    // @invariant 0 <= red, green, blue <= 255
    private Color color;

    // @requires h, w > 0
    // @requires 0 <= red, green, blue <= 255
    public ColoredRectangle (int h, int w, int red, int green, int blue) {
        super(h,w);
        color = new Color(red, green, blue);
    }

    public void darken() {
        color.darken();
    }
}
```

3. Considere o diagrama apresentado ao lado, desenvolvido durante a fase de desenho de um projeto.

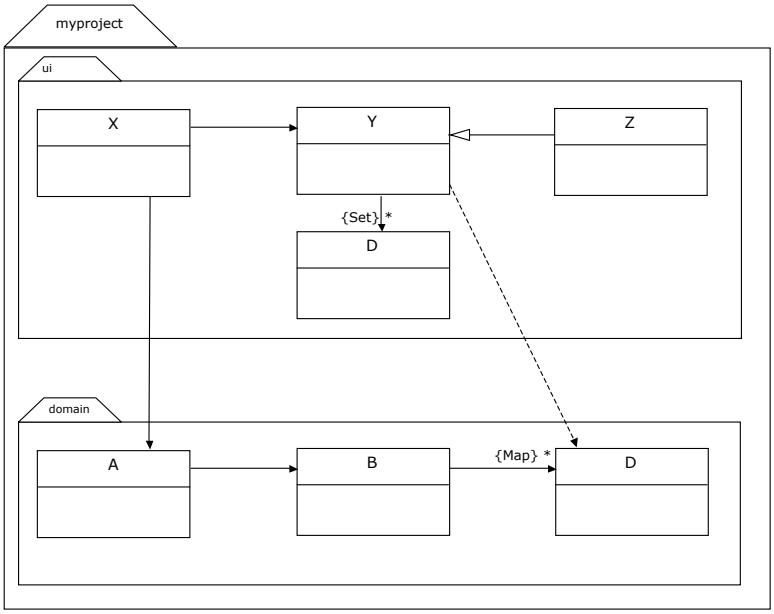
a) Esboce o modelo de implementação correspondente a este diagrama. Justifique convenientemente as opções que tomar relativamente ao controlo de acesso.

b) De acordo com o seu modelo de implementação, de que forma se pode referir, na classe `Y` de `ui`, à classe `D` de `domain` e à classe `D` de `ui`? Indique, por exemplo, de que forma definiria em `Y` um método que recebe um objeto da primeira classe e devolve um objeto da segunda.

c) Se a classe `Y` de `ui` fosse definida como pública e incluísse os seguintes métodos

```
public void m() {...}
protected void n() {...}
void k() {...}
private void l() {...}
```

em que outras classes do projeto estes métodos poderiam ser usados?



4. Considere as classes na figura à direita.

a) Para cada um dos fragmentos de código apresentados abaixo verifique se:

- código não compila,
- o código compila com *warnings*,
- o código gera erros em tempo de execução

```
1.  AnimalHouse<Animal> h1 = new AnimalHouse<Cat>();

2.  AnimalHouse<Dog> h2 = new AnimalHouse<Animal>();

3.  AnimalHouse<?> h3 = new AnimalHouse<Cat>();
    h3.setAnimal(new Cat(4));

4.  AnimalHouse h4 = new AnimalHouse();
    h4.setAnimal(new Dog(4));
```

b) Programe um método que, dada uma casa de animais (*Animal*), calcula o número de pernas do seu habitante.

c) Programe um método que, dado um *AnimalHouse* e um *Animal*, coloque o animal dentro da casa.

```
public class AnimalHouse<E> {
  private E animal;

  public void setAnimal(E x) {
        animal = x;
  }
  public E getAnimal() {
        return animal;
  }
}

public class Animal {
  private int legs;

  public Animal(int legs){
        this.legs = legs;
  }
  public int getLegs(){
        return legs;
  }
}

public class Cat extends Animal {
  public Cat(int legs){
        super(legs);
  }
…
}

public class Dog extends Animal {
  public Dog(int legs){
        super(legs);
  }
…
}
```

5. Relembre os tipos de dados implementados no exercício 1 da Série 5. Considere a classe embaixo:

a) Diga, indicando o respetivo número de linha, quais as instruções incorretas no `main`, e explique porquê.

b) Construa um método `static addCandidate` que acrescente um dado candidato a cada concurso de uma dada coleção de concursos, desde que esse candidato seja elegível para esse concurso. Apresente uma solução que não provoque erros de compilação quando invocado através das seguintes instruções:

```
addCandidate(c1,l1);
addCandidate(c1,l2);
addCandidate(c1,l3);
addCandidate(c1,l4);
```

```
1  public class AlineaA {
2
3      public static void main(String[] args){
4
5          MyKindOfContest myCont = new MyKindOfContest(
6                                   LocalDateTime.of(2021, 12, 12, 16, 30),1,5);
7          myCont.addCandidate("Rita", new int[]{3,4,3,5,3,5});
8          myCont.addCandidate("Pedro", new int[]{5,5,5,4,4,3});
9
10         FlexibleCriteriaContest yourCont = new FlexibleCriteriaContest(
11                                       LocalDateTime.of(2021, 12, 12, 16, 30),
12                                       1,5);
13         yourCont.addCandidate("Rita", new int[]{3,4,3,5,3,5});
14         yourCont.addCandidate("Pedro", new int[]{5,5,5,4,4,3});
15
16         Contest hisCont = new MyKindOfContest(
17                                    LocalDateTime.of(2021, 12, 12, 16, 30),1,5);
18         hisCont.addCandidate("Rita", new int[]{3,4,3,5,3,5});
19         hisCont.addCandidate("Pedro", new int[]{5,5,5,4,4,3});
20
21         List<MyKindOfContest> l1 = new ArrayList<MyKindOfContest>();
22         l1.add(myCont);
23         l1.add(hisCont);
24
25         List<Contest> l2 = new ArrayList<MyKindOfContest>();
26         l2.add(myCont);
27
28         List<Contest> l3 = new LinkedList<Contest>();
29         l3.add(myCont);
30         l3.add(yourCont);
31         l3.add(new Contest(LocalDateTime.of(2021, 12, 12, 16, 30),1,5));
32
33         List<? extends Contest> l4 = l3;
34         for(Contest cont : l4){
35               System.out.println(cont.maxGrade());
36         }
37         l4.add(hisCont);
38     }
39 }
```