Princípios do Projeto de Classes

Fernando Magno Quintão Pereira

September 1, 2010

Questão 1 – OCP

A companhia de brinquedos *Toy Inc.* precisa de um sistema para controlar sua folha de pagamento. Esta companhia possui três categorias de funcionários: gerentes, vigias e demais funcionários. O salário de cada tipo de funcionário é calculado pela seguinte formúla:

- Se o funcionário pertence à categoria demais funcionário, então seu salário é igual ao número de horas trabalhadas multiplicado por R$ 12,00.
- Se o funcionário pertence à categoria vigia, então seu salário é igual ao número de horas trabalhadas multiplicado por R$ 16,00 mais R$ 80,00 vezes o fator de periculosidade. Este é um valor entre 1.0 e 10.0.
- Se o funcionário pertence à categoria gerente, então seu salário é igual ao número de horas trabalhadas multiplicado por R$ 20,00, mais R$ 100,00 vezes o número de funcionários subordinados a ele.

Questão 2 – OCP

A nossa companhia *Toy Inc.* acaba de definir um quarto tipo de funcionário: trata-se do *funcionário super-produtivo*. Este tipo de funcionário também recebe pela hora trabalhada, porém ele ganha um pouco mais: R$ 18,00 por hora, mais uma bonificação de R$ 80,00. Que modificações precisam ser feitas sobre o software para tratar este novo tipo de empregado?

Questão 3 – OCP

São tempos de crise... *Toy Inc.* está passando um aperto, e é preciso priorizar: Empregados super-produtivos precisam receber primeiro. E se houver dinheiro sobrando, então a empresa paga os demais funcionários. Escrever o método `getLuckyList` que deve receber, além da coleção de funcionários, também um total de dinheiro que pode ser gasto em seu pagamento, e retorne a lista dos empregados que serão pagos.

Questão 4 – OCP

Será que é possível garantir o fechamento contra ordenação? Será que a criação de um novo tipo de funcionário sempre nos forçará a ter de modificar o método que paga os funcionários em ordem?

Questão 5 – OCP

Eu gostaria de contar o número de guardas na folha de pagamento. Neste caso vale a pena usar introspecção. Isto contraria o princípio da abertura-fechamento?

Questão 6 – LSP

Utilize os dois módulos abaixo para implementar uma classe `Square`.

```
public class Rectangle implements Shape {
  private int x, y, w, h;
  public Rectangle(final int xp, final int yp,
                   final int wp, final int hp){
    this.x = xp; this.y = yp;
    this.w = wp; this.h = hp;
  }
  public void setWidth(int newW) {
    this.w = newW;
  }
  public void setHeigth(int newH) {
    this.h = newH;
  }
  public int area() {
    return w * h;
  }
  public final void draw(final Graphics c) {
    c.drawRect(x, y, w, h);
  }
}
```

Questão 7 – LSP

Como lidar com um programa deste tipo?

```
public class TestRect {
  static void testArea(Rectangle r, int h, int w)
  throws Exception {
    r.setHeight(h);
    r.setWidth(w);
    if (r.area() != h * w) {
      throw new Exception("Ops!");
    }
  }
  public static void main(String args[]) throws Exception {
    testArea(new Rectangle(50, 50, 10, 20), 20, 40);
    testArea(new Square(50, 50, 20), 20, 40);
  }
}
```

Questão 8 – LSP

Lembra-se da nossa classe `MyList` da última aula? Estenda esta classe para que ela passe a ter acesso persistente. Isto é, ela deve sempre salvar os dados armazenados em um arquivo.

Questão 9 – LSP

Algum problema neste programa?

```
import java.util.*;

public class TestRect2 {
  public static void getMockData(Collection<Rectangle> c) {
    c.add(new Rectangle(5, 50, 10, 20));
    c.add(new Rectangle(50, 5, 10, 20));
    c.add(new Square(50, 5, 10));
    c.add(new Square(51, 5, 10));
  }

  public static void main(String args[]) throws Exception {
    Collection<Rectangle> c = new TreeSet<Rectangle>();
    getMockData(c);
    for (Rectangle r : c) {
      System.out.println(r);
    }
  }
}
```