Características das Linguagens de Programação - P2 – 31/01/2014

1ª questão (2 pontos): Considere os estacionamentos para carros. Você pode modelar estacionamentos em java através da classe *Estacionamento*. Esta classe possui uma variável de instância *vagas* que informa sobre o número de vagas disponível em um dado estacionamento e dois métodos *entra* e *sai*, que são invocados por objetos do tipo *Carro* (a contrapartida dos carros do mundo "real"), sempre que um carro entra ou deixa o estacionamento. O código, incompleto, da classe *Estacionamento* é transcrito a seguir:

```
class Estacionamento {
        private int vagas;
        public Estacionamento (int vagas) { // construtor
                ...
        }

        ... entra () { // entra no estacionamento
                ...
        }

        ... sai () { // deixa o estacionamento
                ...
                }
}
```

Complete o código da classe *Estacionamento*, de forma a simular o uso concorrente de um estacionamento por vários carros.

2ª questão (2 pontos): Carros, por sua vez, podem ser modelados pela classe *Carro*. Uma instância (objeto) da classe *Carro* pode ser visto como um *thread*. Um carro percorre as ruas durante um certo tempo (este percurso pode ser implementado via o método *sleep()*), antes de entrar em um estacionamento *p*. Pode-se também usar o método *sleep()* para simular o tempo que o carro permanece no estacionamento *p* antes de sair do mesmo. A classe *Carro* possui duas variáveis de instância: i) a sua placa, que pode corresponder ao nome do *thread;* ii) um objeto do tipo Estacionamento, que indica em que estacionamento o carro irá estacionar. O código, incompleto, da classe *Carro* é transcrito a seguir:

```
class Carro extends Thread {
        private Estacionamento p;
        public Carro(String placa, Estacionamento p) {
                super(placa);
                this.p = p;
                start();
        }

        public void run() {
                ...
}
```

Complete o código da classe *Carro*, de forma a simular o comportamento dos carros, conforme descrito anteriormente. Imprima mensagens indicativas quando um carro entrar ou sair do estacionamento.

3ª questão(2 pontos): Crie uma classe denominada EstacionandoCarros que crie o objeto *central*, do tipo *Estacionamento*, com 5 vagas. A seguir, instancie 40 carros, que depois de percorrer as ruas por um certo tempo (aleatório), irão procurar entrar no estacionamento *central*, estacionar por um dado tempo (aleatório) e, enfim, deixar o estacionamento não mais retornando ao mesmo.
Observação: As placas dos carros são diferentes entre si e é admissível qualquer combinação de letras e/ou número para gerá-las.

4ª questão(2 pontos): Considere que, durante o ano de 2013, você desenvolveu uma interface denominada *EnginePlus2013* que passou a ser implementada por mais de 200 programadores em aplicativos por eles desenvolvidos para a corporação em que você trabalha.

```
Public interface EnginePlus2013{
        void executeAction_01(int i, double x);
         int executeAction_02(String s);
         …
         float executeAction_10 (String s1, String s2);
}
```

Posteriormente, em janeiro de 2014, você adicionou mais dois métodos à sua interface, a saber:

```
String[] executeAction_11 (Integer i1, Float f2);
Long executeAction_12 (Boolean b1, Character x);
```

Dos 200 programadores que implementaram *EnginePlus2013*, 20 precisam adicionar de imediato aos seus programas que implementam *EnginePlus2013*, os métodos executeAction_11, executeAction_12. Os outros 180 programadores não precisarão jamais fazer uso destes métodos nos programas que já implementam *EnginePlus2013*. No entanto, sabe-se que quase todos precisarão incluir os métodos executeAction_11, executeAction_12 nos novos programas desenvolvidos a partir de janeiro de 2014. Qual a solução que você daria, em Java, para evitar que os 180 programadores alterem os programas que implementam *EnginePlus2013*? Escreva o código correspondente em Java.

5ª questão (2 pontos): Considere a *public class* ***LinkedList<E>*** constante da ***Java™ Platform Standard Ed. 7***. Empregando o conceito de *Generics*, desenvolva um programa que use ***LinkedList<E>*** para criar duas listas encadeadas, uma, *li*, contendo apenas objetos do tipo *Integer* e outra, *ls*, só objetos do tipo *String*. Adicione objetos apropriados a cada lista. No mesmo programa crie uma lista *lAll* que seja a união das duas listas anteriores. Depois use os métodos de ***LinkedList<E>*** para recuperar o primeiro elemento e o último elemento de cada lista. Não esqueça de tratar possíveis exceções. Por fim mostre os elementos recuperados na tela do seu terminal.

Boa Sorte!
Prof. Oscar Luiz Monteiro de Farias