Universidade de Mogi das Cruzes
Padrões de Projetos Orientados a Objetos - Profª. Danielle Martin

Lista de Exercícios 1

1. Associe os conceitos relacionados à herança na orientação a objetos localizados na coluna da esquerda com as suas definições na coluna da direita:

A) extends	( ) Relacionamento É-UM entre duas classes
B) classe Object	( ) Superclasse de todas as classes do Java
C) hierarquia de classes	( ) Palavra reservada usada para se referir a um membro da superclasse
D) protected	( ) Define membros que podem ser acessados pela própria classe e suas subclasses
E) método toString()	( ) Método da classe Object que imprime uma versão String de qualquer objeto
F) método clone()	( ) Classe mãe em um relacionamento de herança
G) super	( ) Palavra reservada para impedir que um membro da classe seja sobrescrito
H) superclasse	( ) Capacidade de desacoplar a referencia de um objeto de sua implementação
I) subclasse	( ) Palavra reservada que indica que uma subclasse irá herdar de uma superclasse
J) final	( ) Método da classe Object que cria uma cópia do objeto que o chamou
K) herança	( ) Um grupo de classes relacionadas em vários níveis de herança
L) polimorfismo	( ) Classe filha em um relacionamento de herança

2. Uma rede de lojas é conhecida por realizar promoções com frequência. Cada promoção define um valor de desconto que pode ser abatido de uma compra. Existem dois tipo de promoções, a promoção que é valida até uma data pré determinada, e a promoção com limite de usos - após X clientes usarem o desconto, a mesma se encerra. Em ambos o casos, cada vez que uma compra tentar usar um desconto de promoção, deve-se verificar se a mesma ainda está ativa antes de aplicar o desconto.

Com base na descrição acima, modelou-se a seguinte hierarquia entre as classes Promoção (genérica) e PromocaoLimiteData e PromocaoLimiteQuantidade (específicas). Os objetos Promoção são usados na classe Venda para que o total seja calculado já com o desconto, se aplicável. Implemente o código das classes em questão e uma aplicação que crie objetos de teste.

Dica: Utilize o método after da classe Date para saber se uma data é maior que outra:

```
Date data1 = new SimpleDateFormat("dd/MM/yyyy").parse("15/03/2016");
Date data2 = new Date();
if (data1.after(data2) ) { //data1 é posterior a data2 }
```