Quis 14 – Classes e Objectos I (14/10/2014)
Versão 2

1. Na escrita de um método sem retorno, quais das instruções são permitidas no seu corpo?
C. return;
D. omissão da instrução return.

2. Qual o modificador que indica que não existem restrições de acesso à propriedade ou ao método?
A. public;

//**public**: Qualquer objeto pode aceder ao membro;
//**default**: Qualquer objeto do mesmo pacote pode aceder o membro e subclasses de outros pacotes;
//**protected**: O membro é acessível apenas por objetos do mesmo pacote;
//**private**: O membro é acessível apenas internamente (próprio objeto);

3.

```
public class Person {
   String name;
   public int id;
   public Person(String name, int id) { this.name = name; this.id = id;  }
   (Construtor)
   private String getName() { return name; }
   public void setName(String name) { this.name = name; }
   public int getId() { return /*COLOQUE O CODIGO AQUI*/; };
}

public class App {
   public static void main(String[] args) {
       Person person = new Person("Luis", 14);
       person.id = 14; //retirar
   }
}
```

B. Alterar o modificador do método *getName* para *public*;
C. Receber a variável *id* no construtor e fazer a afectação *this.id = id*;
E. Alterar o modificador das propriedades *id* e *name* para *private*.
F. Alterar o /*COLOQUE O CODIGO AQUI*/ por *this.id*;

//Os métodos **getters** e **setters** são responsáveis por fornecer meios de acedermos e modificarmos valores dos atributos de um objeto.
A convenção para estes métodos é a seguinte:
- **Getters**: Método que retorna o atributo, é sempre composto pela palavra *get[nome do atributo]*. Ex: getIdade(), getSalario().
- **Setters**: Método que atribui/modifica o valor de um atributo, é sempre composto pela palavra *set[nome do atributo]* e o parâmetro do mesmo tipo do atributo. Ex: setIdade(Integer idade), setSalario(Double salario).
São ambos métodos public.

4. Quais das seguintes afirmações são benefícios da utilização do encapsulamento?
A. Permite que a implementação da classe modifique, sem que isso afecte os programas que a utilizam; //podemos fazer alterações na classe sem modificar o programa.
D. Permite que a implementação da classe proteja as suas propriedades;
F. Permite a criação segura de várias instâncias da classe.

// As restantes não estão relacionadas com o encapsulamento
B. Protege informação confidencial que esteja guardada nos objectos;

C. Previne que o código lance excepções;
E. Permite que as classes sejam combinadas dentro do mesmo pacote;

5. (A.) 3 objectos.

```
Integer i1 = new Integer(1); //Objecto do tipo Integer com valor 1
Integer i2 = 2;   //Objecto do tipo Integer com valor 2
Integer i3 = new Integer(3); //Objecto do tipo Integer com valor 3
```