Exame: Quiz #05 **Aluno:** Rodrigo João Bastos Mendes

Resultado obtido não conta para a sua avaliação.
Apenas uma das respostas está correcta por pergunta.
Podem ser escolhidas várias respostas a uma mesma pergunta. Pode tornar a responder quantas vezes quiser ao questionário.
O tempo mostrado é só para uma tentativa de resposta a todo o questionário.
Para efeitos informativos, é mostrada uma pontuação: Cada questionário estará online para efeitos de frequência durante um curto perí
1 resposta correta: 1 pontos
1 resposta errada: -0.5 pontos

Valorização: 10
Correcção: as afirmações correctas estão marcadas com C e as erradas estão marcadas com E

Sair

1. TAD Conjunto

Na API do próprio Java, um **Set** é o quê?

[ ] E ~~Uma classe abstracta~~
[x] C **Um interface**
[ ] E ~~Uma classe "normal"~~

2. Genéricos I

Quando queremos ter uma única implementação de uma classe que suporta vários tipos de objectos podemos usar **genéricos**.
Qual das seguintes declarações estaria correcta para **declarar um classe XPTO para um tipo genérico T**?

[x] C **class XPTO<T>**
[ ] E ~~class XPTO[T]~~
[ ] E ~~class XPTO(T)~~

3. Genéricos II

O que acontece quando são compiladas as seguintes linhas:
Integer fortytwo = 42;
int i = fortytwo;

[ ] E ~~A linha compila sem erros, porque o número 42 é a resposta sobre o sentido da vida (pesquisar no google sobre "what is the meaning of life universe and everything")~~
[ ] E ~~A linha dá um erro de compilação porque não podemos colocar um tipo primitivo a referenciar um objecto, sendo que falta fazer um *cast*~~
[x] C **A linha compila sem erros, porque o Java faz *auto unboxing* do número contido no objecto do tipo Integer**

4. Herança I

Para que serve o mecanismo de **herança**?

[x] C **Para criar uma hierarquia de classes, com subclasses a herdarem métodos e atributos de outras classes**
[ ] E ~~Para permitir que qualquer variável tenha um valor padrão que é herdado a partir do definido no padrão da linguagem~~
[ ] E ~~Para que seja possível herdar um parâmetro quando estamos a implementar um método recursivo~~

5. Herança II

Imagine que tem duas classes com as seguintes linhas iniciais:
class Tipo1 {...}
class Tipo2 extends Tipo1 {...}

Considere duas linhas a colocar numa outra classe:
(i) Tipo1 t = new Tipo2();
(ii) Tipo2 t = new Tipo1();

No que toca a erros de compilação, o que acontece?

[ ] E ~~ambas as linhas dariam erro de compilação~~
[x] C **(i) seria compilada e (ii) daria erro de compilação**
[ ] E ~~(i) daria erro de compilação e (ii) seria compilada~~

6. Herança III

Imagine que tem as três seguintes classes:

```
class TipoA {
public void write() {System.out.println("Sou do TipoA");}
}

class TipoB extends TipoA {
public void write() {System.out.println("Sou do TipoB");}
}
```

```
class TipoC extends TipoB {
}
```

Considere que tem as seguintes duas linhas:
TipoC c = new TipoC();
c.write()

O que acontece?

[ ] E ~~É gerado um erro de compilação, porque o TipoC não tem o método write()~~
[ ] E ~~É escrito "Sou do TipoA"~~
[x] C **É escrito "Sou do TipoB"**

7. Herança IV

Qual é a classe na **raíz da hierarquia de todos os objectos** de Java?

[ ] E ~~java.lang.Root~~
[x] C **java.lang.Object**
[ ] E ~~java.lang.Top~~

8. Listas Ligadas I

No contexto de uma lista ligada, o que é um **nó**?

[ ] E ~~É um atributo da lista ligada contendo um valor~~
[ ] E ~~É uma referência para o próximo elemento da sequência~~
[x] C **É um "elemento" da sequência, armazenando um valor e uma referência para o próximo elemento da sequência**

9. Listas Ligadas II

Tal como num array, numa lista ligada, os elementos da lista estão sempre em posições contíguas de memória. Esta afirmação é **verdadeira ou falsa**?

[x] C **falsa**
[ ] E ~~não é possível responder~~
[ ] E ~~verdadeira~~

10. Listas Ligadas III

Suponha que um nó *node* de uma lista ligada tem atributos *value* e *next* com o significado óbvio. O que faz a instrução seguinte? (supondo que *node* não é o último nó da lista e que o atributo *next* é publico)
node.next = node.next.next;

[ ] E ~~um programa com esta instrução daria um erro de compilação~~
[ ] E ~~remove o nó *node*~~
[x] C **remove o nó seguinte a *node***