Instituto Politécnico de Castelo Branco

Escola Superior de Tecnologia.
Curso: Engª Informática + Tecnologias da Informação e Multimédia
2º Semestre do 1º Ano
Unidade Curricular: Programação II
Docente: Fernando Sérgio Barbosa
Ano Letivo 2015/2016
Época Recurso
Data: 18/7/2016 Duração: 2h00 sem tolerância

Classificação das questões:

Questões	1	2			3			4			
Alíneas		1	2	3	1	2	3	1	2	3	4
Classificação	13x0,4 (-0,1 errada)	1,5	1,5	1,5	1,5	1	2	1	1,6	1,6	1,6

Nome:_____Número:_____

Curso:_____

1. Das seguintes afirmações escolha **a correta** (as erradas descontam 0.1). RESPONDA NO ENUNCIADO.

1.1. Considere as coordenadas polares. Cada coordenada é definida pela distância do ponto à origem e o ângulo que faz com a horizontal. A distância nunca será negativa e o ângulo tem de ser um valor entre 0 e 360. As variáveis que seriam necessárias na classe CoordPolar seriam :

- ❑
```
int distancia;
char angulo;
```
- ❑
```
float distancia;
float angulo;
```
- ❑
```
Distancia dist;
Angulo ang;
```
- ❑
```
String distancia;
String angulo;
```

1.2. Em Java, os níveis de acesso são aplicáveis:
- ❑ apenas às variáveis.
- ❑ apenas às variáveis e a métodos.
- ❑ às classes, variáveis e métodos.
- ❑ às variáveis, métodos e packages.

1.3. Considere o seguinte excerto de código:

```
Recurso( int x, int y ) {
        this.x = x;
        this.y = y;
}
```

```
Recurso( int x ) {
    this( x, 0 );
}
```

```
Recurso( ) {
    this( 0, 0 );
}
```

1.3.1. Os métodos apresentados:
- ❑ são construtores
- ❑ são setters.
- ❑ são getters.
- ❑ são métodos de implementação.

1.3.2. Os métodos apresentados pertencem à classe:
- ❑ Object.
- ❑ Integer.
- ❑ Recurso.
- ❑ impossível de saber.

1.3.3. No método Recurso(int x, int y) o this está a referir-se:
- ❑ ao objeto a ser usado.
- ❑ a um construtor.
- ❑ à variável this da classe.
- ❑ à própria classe.

1.3.1. No método Recurso(int x) o this está a referir-se:
- ❑ ao objeto a ser usado.
- ❑ a um construtor.
- ❑ à variável this da classe.
- ❑ à própria classe.

1.4. O estado de um objeto:

- ❑ é variável.
- ❑ é dependente da sua identidade.
- ❑ depende do seu nível de acesso.
- ❑ influência a sua identidade.

1.5. A relação entre um livro e a sua capa é melhor representada por:

- ❑ herança.
- ❑ agregação.
- ❑ associação.
- ❑ uso.

1.6. Para indicar que uma classe não pode ter subclasses:

- ❑ declara-se a classe como static.
- ❑ declara-se a classe como final.
- ❑ declara-se a classe como const.
- ❑ declara-se a classe como noextends.

1.7. Os métodos declarados numa interface:

- ❑ podem ter qualquer nível de acesso.
- ❑ podem ter qualquer nível de acesso menos private.
- ❑ têm de ser públicos ou protected.
- ❑ têm de ser públicos.

1.8. Considere o seguinte código

```
1  static int metodoUm( ){
2     int x = 10;
3     try {
4        rebenta();
5     }
6     catch( NullPointerException e){
7        x = 30;
8     }
9     catch( InvalidArgumentException e ){
10       x = 40;
11       throw e;
12    }
13    return x;
14 }
15 public static void main( String []args ){
16    int res = metodoDois();
17    System.out.println("res = " + res );
   }
```

```
18 static int metodoDois(){
19    int x = 100;
20    try {
21       x = metodoUm();
22    }
23    catch( NullPointerException e){
24       x = 120;
25    }
26    catch( NumberFormatException e ){
27       x = 130;
28    }
29    return x;
30 }
31
```

1.8.1. Considerando que o método rebenta() atira a exceção NumberFormatException o código imprime:

- ❑ res = 10.
- ❑ res = 130.
- ❑ res = 100.
- ❑ uma mensagem de erro.

1.8.2. Considerando que o método rebenta() atira a exceção NullPointerException o código imprime:

- ❑ res = 100.
- ❑ res = 120.
- ❑ res = 30.
- ❑ uma mensagem de erro.

1.8.3. Considerando que o método rebenta() atira a exceção InvalidArgumentException o código imprime:

- ❑ res = 100.
- ❑ res = 10.
- ❑ res = 40.
- ❑ uma mensagem de erro.

2. Recorde a classe CoordPolar descrita na alínea 1.1.

2.1. Implemente a classe CoordPolar, declarando os getters e setters. Deve usar a exceção IllegalArgumentException.

2.2. Implemente o método toString para esta classe, de forma a escrever a coordenada no formato "r=*distância* a=*angulo*".

2.3. Crie os métodos getX() e getY(), que retornam, em coordenadas cartesianas, o valor da coordenada polar. As fórmulas de conversão são:

x = distância * cos(ângulo) y = distância * sin(ângulo)

3. Considere as seguintes classes, as quais fazem parte de um sistema de avaliação de estabelecimentos. Em cada avaliação, o estabelecimento recebe uma classificação de 1 a 5 estrelas. Cada estabelecimento pode pertencer a uma ou mais categorias (restaurante, hotel, etc) .

```
class Avaliacao {
   int nEstrelas;
   String descricao;
}
class Estabelecimento {
   String nome;
   Vector<Categoria> categorias = new Vector<Categoria>();
   CoordPolar coord;

   float getMediaAvaliacao() { ... }
}
class Categoria {
   String nome;
   String descricao;
   Vector<Estabelecimento> estabelecimentos = new Vector<Estabelecimento>();
}
class SistemaAvaliacao {
   Vector<Estabelecimento> estabelecimentos;
   Vector<Categoria> categorias;

   Vector<Estabelecimento> getEstabelecimentosProximos( CoordPolar cp,
                                                        Categoria cat, float distancia ){...}
}
class CoordPolar {
   float distancia( CoordPolar cp ){ ... }
}
```

3.1. Elabore o diagrama de classes deste sistema. Não coloque variáveis nem métodos.

3.2. Complete o seguinte método, que retorna um vetor de estabelecimentos que pertencem a uma dada categoria e estão a uma distância menor que a indicada de uma dada coordenada. Para isso deve preencher cada espaço com **uma única** palavra. RESPONDA NO ENUNCIADO.

```
Vector<Estabelecimento> getEstabelecimentosProximos(CoordPolar cp,
                                    String cat, float distancia ){
   Categoria categ = null;
   for( Categoria c : _____________________ )
      if( c.getNome()._____________________( cat ) ){
          categ = c;
          break;
      }
   if( categ == null )
      return null;

   Vector<Estabelecimento> dentro = new Vector<Estabelecimento>();
   for( _____________________ e : categ.getEstabelecimentos() )
      if( e.getCoord().distancia( cp ) <  _____________________ )
          dentro.add( e );
   return _____________________;
}
```

3.3. Implemente o método `float getMediaAvaliacao()` da classe Estabelecimento que retorna a média de todas as avaliações do estabelecimento.

4. Considere as seguintes classes.

```
public class ClasseAAA {

    private int a, b;

    public int m1( int x) {
        return  x + a;
    }

    public int m2( ) {
      return b + a;
    }

    public int m3( int x) {
        return  m2() + x;
    }
}
```

```
public class ClasseBBB {

    private int a, b, c;

    public int m1( int x) {
        return  x + a;
    }

    public int m2( ) {
      return b + a + c;
    }

    public int m3( int x) {
        return  x + c;
    }
}
```

```
public class ClasseCCC {

   private int a, b, d;

   public int m1( int x) {
      return  x + a;
   }

   public int m2( ) {
     return b + a;
   }

   public int m3( int x) {
      return  x + b + d;
   }
}
```

```
public void ClasseDDD {

   public void metodo( Object o, int tipo ) {
      switch( tipo ){
       case TIPO_AAA: mA( (AAA) o ); break;
       case TIPO_BBB: mB( (BBB) o ); break;
       case TIPO_CCC: mC( (CCC) o ); break;
      }
   }

   private void mA( AAA a ) {
      a.m1( 2 );
      a.m2( );
      a.m3( 3 );
   }

   private void mB( BBB b ) {
      b.m1( 2 );
      b.m2( );
      b.m3( 3 );
   }

   private void mC( CCC c ) {
      c.m1( 2 );
      c.m2( );
      c.m3( 3 );
   }
}
```

O código anterior tem muito código repetido e o uso de herança seria uma boa ajuda para diminuir essa repetição ao máximo.

4.1. Implemente a interface.

4.2. Implemente a super classe.

4.3. Implemente a classe ClasseAAA.

4.4. Implemente a classe ClasseDDD.