Laboratório 09 - Classes e Objectos I (14/10/2014)

1.

```
package sessão10_10;

//exercício Lab09p1
public class Shirt {
	private String color;
	private String size;
	public Shirt(String color, String size){  //construtor
		this.color = color;
		this.size = size;
	}
	public String getColor(){
		return color;
	}

	public String getSize(){
			return size;
	}
	public void setColor(String color){
		this.color = color;
	}

	public void setSize(String size){
		this.size = size;
	}

	public String print(){
		return this.color + " " + this.size;
	}
}

public class Applab09 {
	public static void main(String [] args){
	   Shirt s = new Shirt("blue", "M");      //um novo objeto, s, shirt com as caracteristicas "blue", "M"
	   Shirt s1 = new Shirt("yellow", "L");    //um novo objeto, s1, shirt com as caracteristicas "yellow", "L"
	   s1.setSize("XL");   // modificar tamanho do objeto referenciado por s1 para XL
	   s1 = s;     // o apontador s1 é agora referência para o mesmo objecto que s, ou seja, "blue", "M"
	   s.setColor("red");    // modificar a cor do objeto referenciado por s e s1 para red
	   s1.setSize("S");     // modificar o tamanho do objeto referenciado por s1 e s para s
	   System.out.println(s.print());   // imprime red S
	   System.out.println(s1.print());   // imprime red S   porque s e s1 apontam para o mesmo objeto
	}
}
```

Output:

```
red S
red S
```

2. .

```
package lab09p2;

import java.util.Scanner;
```

```
//exercício lab09 p2
public class StringUtils {
/**
* Counts the number of times that a char occurs in the original
string
* @param original The original sentence
* @param c The char to find in the sentence
* @return The number of times the char c occurs in the sentence
*/
       public int countLetters(String original, char c) {
          int count =0 ;
          int size = original.length();
          for (int i = 0; i < size; i++)
          {
              if(original.charAt(i)== c )      // verifica a igualdade entre os caracteres
da string e o 'c'
                  count++;     // se for igual incrementa a contagem
          }
              return count;
       }

       /**
       * Indicates if the original sentence contains another sentence
       * @param original The original sentence
       * @param toFind The sentence to find
       * @return A string with the result
       */
       public String exists(String original, String toFind)
       {
          int size = original.length();   // tamanho da String Original
          int toFindSize = toFind.length();   // tamanho da string toFind
          String message = "";
          String auxString = "";

          if (size<toFindSize) //verificar se o tamanho da string a encontrar, é menor
que o tamanho da string original
          {
              message = toFind + " does not exist in " + original;
          }
          else{
              for (int i=0; i<size-toFindSize+1; i++)  //não é necessário verificar a
string original toda
              {
                 if (original.charAt(i) == toFind.charAt(0)) // o
                 {
                     auxString = original.substring(i, i+toFindSize); //gerar uma
substring da string original a partir do carater encontrado, com o tamanho da string
toFind

                     if (auxString.equals(toFind)) // verificar se a string que obtivemos
é igual à string toFind
                     {
                        message = toFind + " exists in " + original;
                        break;
                     }
                 }
              }
          if (message==""){  //se não houver qualquer correspondência escreve a seguinte
mensagem
              message = toFind + " does not exist in " + original;
          }
       }
       return message;
   }
```

```
	/**
	* Sorts the chars in a the original sentence
	* @param original The original sentence
	* @return The string sorted
	*/
	public String sort(String original) {   //retorna a sequência ordenada
	   int size = original.length();
	   char[] originalArray = new char[size];

	   for (int i = 0; i < size; i++)
	   {
	      originalArray[i] = original.charAt(i); //colocação dos caracteres da string
num array
	   }

	//método BubbleSort
	   for(int i = 0; i < size; i++) // i representa o número de elementos já
ordenados
	   {
	      for(int j = 1; j < size - i; j++)  // j índice do elemento que está a ser
comparado
	      { //trocas até size - i porque a cada iteração, eu tenhos os i elementos
maiores já ordenados
	         if(originalArray[j] < originalArray[j - 1])
	         {
	            char exchangeAux = originalArray [j-1];    //colocar na variável de
apoio o valor da variável que está a ser testada
	            originalArray [j-1] = originalArray[j];   //colocar no índice da
variável que está a ser testada o valor do índice que é mais baixo
	            originalArray[j] = exchangeAux;
	         }
	            }
	      }

	      String orderedMessage = "";

	      for (int i = 0; i < size; i++)
	      {
	            orderedMessage += (char)originalArray[i];  //colocação dos elementos
já ordenados do array numa String
	      }
	      return orderedMessage;
	}


	/**
	* Returns a String that is a substring of the original sentence
	* @param original The original sentence
	* @param start The index that starts the cut
	* @param end The index that ends the cut
	* @return The substring of the original sentence, from start to
	end
	*/
	public String cutFrom(String original, int start, int end) {

	   return original.substring(start, end);
	}

public class App {
	public static void main(String [] args){
	   StringUtils su = new StringUtils();
	   System.out.println(su.countLetters("Challenge", 'l'));
	   System.out.println(su.exists("Formação Randstad", "stad"));
	   System.out.println(su.sort("exhjevsa"));
	   System.out.println(su.cutFrom("Java Programming", 5, 12));
	}
```

```
}
       public static void main(String[] args) //alternativo
       {
           Scanner input = new Scanner (System.in);
           System.out.println("Insira o texto a analisar");
           String original = input.nextLine();
           System.out.println("Que letra pretende contar");
           char c = input.next().charAt(0);   // da String recolhe apenas o caracter da
primeira posição
           System.out.println("Qual a palavra que pretende pesquisar");
           input.nextLine();     //Limpar o nextLine
           String toFind = input.nextLine();
           System.out.println("string: start");
           int start  = input.nextInt();
           System.out.println("string: end");
           int end  = input.nextInt();

           StringUtils su = new StringUtils();
           System.out.println(su.countLetters(original,c));
           System.out.println(su.exists(original, toFind));
           System.out.println(su.sort(original));
           System.out.println(su.cutFrom(original, start, end));
       }
}
```

Output (alternativo)

```
Insira o texto a analisar
Formação Randstad
Que letra pretende contar
o
Qual a palavra que pretende pesquisar
stad
string: start
5
string: end
10
2
stad exists in Formação Randstad
 FRaaaddmnoorstãç
ção R
```

3. .

```
package Lab09p3;

import java.util.Scanner;

public class Point {
       public double abcissa;
       public double ordenada;

       public Point(double x, double y){    //construtor
           this.abcissa = x;
           this.ordenada = y;
       }

       public  Point addAPointToThePoint(Point p){     //Adiciona ao ponto as coordenadas
de p;
           double x = (this.abcissa + p.abcissa);        //somar as abcissas
           double y = (this.ordenada + p.ordenada);    // somar as ordenadas
           return new Point(x,y);
       }
```

```
        public boolean isTheSamePoint(Point p){  //Indica se o ponto p é igual ao ponto
(têm a mesma abcissa e a mesma ordenada);
            return(haveTheSameAbscissa(p)&&haveTheSameOrdinate(p))?(true):(false);
        }

        public boolean haveTheSameAbscissa(Point p){     //Indica se têm a mesma abscissa;
            return (this.abcissa == p.abcissa)?(true):(false);
        }

        public boolean haveTheSameOrdinate(Point p){     //Indica se têm a mesma ordenada;
            return (this.ordenada == p.ordenada)?(true):(false);
        }

        public Point getBetweenPoint(Point p){     //Retorna um ponto intermédio;
            double x = (this.abcissa + p.abcissa)/2;         // Pm = ((x1+x2)/2; (y1+y2)/2)
            double y = (this.ordenada + p.ordenada)/2;

            return new Point(x , y);
        }



        public static void main(String[] args) {
            Scanner input = new Scanner (System.in);
            System.out.println("Insira as coordenadas dos pontos p1 e p2:");
            System.out.print("x1=");
            int x1 = input.nextInt();
            System.out.print("y1=");
            int y1 = input.nextInt();
            Point p1 = new Point(x1, y1);  //criação do objecto p1 do tipo Point
            System.out.print("x2=");
            int x2 = input.nextInt();
            System.out.print("y2=");
            int y2 = input.nextInt();
            Point p2 = new Point(x2, y2);    //criação do objecto p2 do tipo Point
            System.out.println("p1 = (" + p1.abcissa + " , " + p1.ordenada + " ) e p2 = ("+
p2.abcissa + " , " + p2.ordenada + " )");

            Point p3 = p1.addAPointToThePoint(p2);
            System.out.println("p1 + p2 = (" + p3.abcissa + " , " + p3.ordenada +" )");
            System.out.println("p1 = p2? " + p1.isTheSamePoint(p2));
            System.out.println("p1 e p2 têm a mesma abcissa? " +
p1.haveTheSameAbscissa(p2));
            System.out.println("p1 e p2 têm a mesma ordenada? " +
p1.haveTheSameOrdinate(p2));

            Point p4 = p1.getBetweenPoint(p2);
            System.out.println("Ponto intermédio entre o ponto 1 e 2 tem coordenadas: ");
            System.out.println("p4 = (" + p4.abcissa + " , " + p4.ordenada +" )");
        }
}
```

Output:

```
Insira as coordenadas dos pontos p1 e p2:
x1=5
y1=-6
x2=2
y2=6
p1 = (5.0 , -6.0 ) e p2 = (2.0 , 6.0 )
p1 + p2 = (7.0 , 0.0 )
p1 = p2? false
p1 e p2 têm a mesma abcissa? false
p1 e p2 têm a mesma ordenada? false
Ponto intermédio entre o ponto 1 e 2 tem coordenadas:
p4 = (3.5 , 0.0 )
```