PG3

Trabalho 1

▼ Grupo1

```
//toString

public String toString() { //de modo a apresentar os dados passados
    return text + " " + "[" + points + "]" + "?" + " " + correctAnswer;
}
```

O método `toString` já se encontra pré definido, em java, quando realizamos este código devemos ter em atenção que estamos a redefini-lo pelo que, pelo que deveríamos usar `@Overide` , de modo a garantir que estamos a fazer uma mudança no código do mesmo.

Neste caso o `toString` irá apresentar no Standard output os valores referentes as variáveis definidas na classe `Query`.

```
//metodo equals

public boolean equals(Object obj ) {
    if(!(obj instanceof Query)){
        return false;
    }
    Query q = (Query) obj; //cast realizado
    return text.equals(q.text)
            && points == q.points
            && correctAnswer.equals(q.correctAnswer);
```

Neste caso aplica-se o mesmo principio que no `toString` ,uma vez que o `equals` também já se encontra definido em java, neste caso o `equals` foi adaptado de acordo com o exercício, o método retorna um booleano e recebe um objeto, como nos necessitamos de efetuar uma verificação referente á classe `Query`, realizamos um cast de modo a afetar a variável `obj` do tipo `Object` para passar a ter as características de uma `Query`.

Antes de efetuar este cast teremos que verificar se este `obj` declarado é uma `Query` ou não ou seja se poderá vir a representar uma e para tal utilizamos o

`instanceof` para este efeito

Com o cast efetuado podemos assim aceder aos campos necessários para as comparações (o `equals` usado para `Strings` e `==` para `ints`).

```
    //2

    //Complete a classe Query, acrescentando:
    //Construtor com três parâmetros o texto, a resposta, e número de pontos da qu
estão.

    public Query(String txt, String ca, int p) {
        text = txt;
        correctAnswer = ca;
        points = p;
    }
```

A classe `Query` neste caso ira receber 3 parâmetros, tendo que se identificar o tipo de cada uma.

De notar que os nomes das variáveis poderá ser o mesmo que os das variáveis globais, mas na chamada das vaiáveis globais teremos que utilizar o `this` que é uma referenciação para as variáveis globais ficaria nessa caso assim: `this.text`.

```
     //Construtor com dois parâmetros o texto e um valor boolean
    //que se a true significa que a resposta correta é “yes” a false significa “n
o”

    public Query(String txt, Boolean bool){ //duvida Perguntar á prof
        text = txt;
        points = 5;
        if(bool == true){ //se o bool for true muda-se o valor do correctAnswer
            correctAnswer = "yes";
            getPoints();
        } else correctAnswer = "no";
    }
```

Este construtor conta com dois parâmetros de entrada sendo eles uma `String` e um `Boolean`. esta função não apresenta nenhum tipo de retorno uma vez que é um construtor e irá utilizar funções que já obtém os valores e que efetuam `return`.

O parâmetro de entrada a `String`, a qual será atribuída á variável E, é importante definir igualmente os pontos da pergunta, de seguida verifica-se se o conteúdo do booleano passado corresponde ao que pretendemos, caso este se verifique verdadeiro afeta-mos o valor da variável `correctAnswer` com o valor de

yes, pois a resposta encontra-se correta, e vamos buscar os pontos da pergunta com o auxilio da função `getPoints`. Caso isto não se verifique a `correctAnswer` é preenchida com *no*, e não existe retorno de valores.

```
    //Os métodos de instância (getters) para obter o texto e o número de pontos
    //Getters

    public String getText() {
        return this.text;
    } //para o texto

    public Integer getPoints() {
        return this.points;
    } //para o numero de pontos

   // public String getCorrectAnswer() {return this.correctAnswer;} //para a respo
sta
```

A classe `Query` recebe neste caso uma `String` e um `Boolean`, atribuir o valor da `String` ao `text`, atribuição do valor a `points`.

Realizar uma condição para o conteúdo do parâmetro `bool` passado, com a devida afetação da variável `correctAnswer` com valor correspondente, utilizar a função `getPoints()` realizada para o retorno do valor dos pontos, estas funções são conhecidas como *Getters*.

```
  //O método de instância compareTo que define a relação de ordem sobre as instânc
ias da classe Query. Sejam q1 e
    //q2 dois objetos do tipo Query e x um valor inteiro tal que x = q1.compareTo
(q2). Se:
    //x<0, significa que o número de pontos da questão q1 é inferior ao número de
 pontos da questão q2;
    //x>0, significa que o número de pontos da questão q1 é superior ao número de
 pontos da questão q2;
    //x==0, significa que o número de pontos da questão q1 é igual número de ponto
s da questão q2.

    public int compareTo(Query q2) {
        //x=q1.compareTo(q2);
        int x; //fator de valor
        x = this.points - q2.points;
        return x;
    }
```

O método `compareTo` apresenta o mesmo funcionamento que o `toString` e o método `equals` na medida em que ele já se encontra definido, o que nos realizamos aqui é um `overide` do método original.

Ele recebe uma `Query`, defini-mos um `int` para guardar o resultado da subtração uma vez que o método retorna um `int`, este subtração será efetuada entre os `points` da questão inicial, para a qual utilizamos o `this` para a sua referenciação, e os `points` do parâmetro passado que é do tipo `Query` tendo assim acesso aos seus campos

O método retorna o valor obtido em x.

```
     // O método estático parse que recebendo por parâmetro uma instância de java.
lang.String retorna a
    //correspondente instância de Query. O formato da string recebida por parâmetr
o é:
    //<param>::= <text> '?' <correct answer> | <text> '[' <points> "]?" <correct a
nswer>
    //Usar os métodos de instância da classe java.lang.String:
    //– int indexOf(int ch, int fromIndex) – para obter os índices dos caracteres
 de separação;
    //– int lastIndexOf(int ch ) – para obter o índice do caractere de separação d
a resposta;
    //– String substring(int beginIndex, int endIndex) – para individualizar as st
rings com o texto, a
    //resposta e o número de pontos;
    //o método estático da classe java.lang.Integer
    // int parseInt(String strNumber) – para obter o valor inteiro correspondente
 aos pontos.

    //A dimensão do int é 32 bits [5]? yes
    //01234567890123456789012345678


   public static Query parse(String s) {
       //fazer a verificaação pelo caracter de separação
            int sp0 = s.indexOf('[', 0);
        if(sp0 == -1) { //caso não exista procuro pelo ?
            //primeira substring com o text
            int sp = s.indexOf('?', 0);
            String text = s.substring(0, sp);

            //segunda substring com a resposta
            String ca = s.substring(sp + 1, s.length());
            return new Query(text,"", 5);

        } else {

            //terceira substring com o text
            String text2 = s.substring(0, sp0);
            text2=text2.trim();

            //quarta substring com o valor de points
            int sp3 = s.lastIndexOf(']');
            String p = s.substring(sp0 + 1, sp3).trim();

            int pi = Integer.parseInt(p); //passar para inteiro
```

```
            //quinta substring com a resposta correta
            int sp4 = s.lastIndexOf('?');
            String ca1 = s.substring(sp4 + 1, s.length()).trim();

            return new Query(text2,ca1,pi);
        }
```

O método `Parse` consiste na fragmentação de uma `String`, `String` essa que é passada como parâmetro ao método.

No inicio é realizado uma verificação na procura de um dos separadores, se o valor for igual a -1, significa que não foi encontrado esse separado pelo que podemos avançar para o próximo sendo este o terminador, será criada uma `String` desde do índice 0 até á posição do separador (contendo esta a pergunta), de seguida é criada outra `String` desde a posição a seguir ao separador até á dimensão final da `String`.

Caso isso não se verifique ou seja é encontrado o primeiro separador, será criada uma`String` desde o inicio até ao separador, será efetuada uma nova pesquisa por outro separador e a segunda `String` será criada desde a posição do primeiro separador mais um ate ao novo separador, de seguida a pesquisa pelo terminador, o mesmo procedimento será efetuado ao referido anteriormente.

De notar que na obtenção da `String` entre os dois primeiros separadores, nós pretendemos esse valor em inteiro e não em `chars`, pelo que usamos o `parseInt` para efetuar essa conversão.

```
     //O método estático quiz que recebendo por parâmetro um parâmetro de dimensão
variável de elementos do tipo
    //Query, instância um Scanner e para cada Query: faz a pergunta; lê a respost
a; e caso esteja correta acumula os
    //pontos. Retorna os pontos acumulados.

   public static Integer quiz(Query[] q){ //retorna me um inteiro numero de pontos
acumulados

       int g = 0; //variavel para acumular o valor dos pontos totais
       Scanner sc = new Scanner(System.in);//criação do scanner
        for(int i = 0; i<q.length;  i++) { //percorrer o array
            if (sc.hasNext() == false) { //se o scanner não estiver a ler mais nad
a é fechado retorna false significa que não tem conteudo
                sc.close(); //fecha o scanner não tem mais nada para ler
            } else if (sc.equals(q[i].correctAnswer)) {//verificar se a resposta e
sta correta e acumula os pontos
                g = g + q[i].points; //soma dos pontos
            }
```

```
        }
       return g;
    }
```

O método `quiz` consiste no retorno do numero total de pontos retornados, tendo em conta as respostas corretas.

Foi declarada um variável de modo a acumular o valor correspondente ao pontos, é instanciado um `scanner`, de modo a ler os inputs, de seguida é inicia-se um ciclo `for` para percorrer o `array` de elementos inseridos até a dimensão do `array` inserido.

É efetuada uma verificação do `scanner` como boa pratica, verificação esta que consiste em ver se o `scanner` se encontra vazio ou preenchido, se tal se verificar o scanner é fechado e a função retorna 0. Caso contrario o conteudo inserido no `scanner` vai comparar a resposta correta com a do `array` inserido com a função `equals`, a resposta for correta acumula os pontos.

A função retorna agora a soma dos pontos.

```
//O método estático growingQueries que recebe por parâmetro um array de questões p
roduz um array ordenado. O
    //array recebido por parâmetro é percorrido sequencialmente e a questão é adic
ionada no fim do novo array se: for a
    //1ª questão; ou se for maior ou igual à última adicionada. Para comparar as Q
uery utilize o método compareTo

    public static Query[] growingQueries(Query[] q){

        int index = 0;//posição do array copia
        Query fn[] = new Query[q.length];//array de copia

        for(int i = 0; i < q.length; i++){ //percorrer o array
                if(i == 0){
                    fn[index]= q[i];
                    index++;
            } else if(q[i].compareTo(fn[index -1]) >= 0){
                    fn[index] = q[i];
                    index++;
            }
        }
        Query copy [] = Arrays.copyOf(fn, index);// redimensionar o tamnho do arra
y
        return copy;
    }
}
```

O método `growingQueries` consiste na criação de um `array` no qual são colocadas os valores das perguntas mais altas.

É iniciado um `index` que será utilizado como o índice do `array` de copia, é criado o `array` de destino com a mesma dimensão do `array` criado.

O ciclo `for` é iniciado até a dimensão do `array` recebido por parâmetro, se o índice for o primeiro é colocado na primeira posição do novo `array`, e o `index` é incrementado, caso contrario será efetuada uma comparação entre a posição seguinte do `array` original e a posição anterior do novo `array` se o valor se verificar superior que o do `array` de copia este valor irá para a posição seguinte do `array`.

Para garantir uma conformidade de dimensões é utilizado o `Arrays.copyOf` para realizar a redimensionalização do `array` final com o tamanho do `index`.

Retorna o array dimensionado.

▼ Grupo2