02

Resolvendo o problema 42 como em uma maratona

Transcrição

Quero mostrar como funciona o meu pensamento quando estou dentro de uma equipe de maratona de programação. Eu vou tentar pensar com as mesmas práticas de antes, extrapolando o que pode acontecer com o meu código e o que o cliente pode ou não fazer. E vou tentar pensar enquanto escrevo.

Além disso, há o fator tempo, que é muito importante em uma maratona. Precisamos prever todas as situações possíveis, mantendo o tempo que usamos para programar mínimo. Cuidado: não estamos falando em minimizar o número de caracteres digitados. Diminuiremos apenas o tempo usado para resolver o problema. Sabemos que o código de uma maratona não será mantido durante anos em uma empresa, então não temos tanto a preocupação de outra pessoa fazer a manutenção do código se ele der algum problema. Mas quanto mais complexo o problema de uma maratona, mais devemos nos preocupar com a qualidade do código, pois ele pode apresentar um erro durante a competição. E seremos nós que teremos de fazer a manutenção para encontrar e corrigir esse erro. Você verá que existe um certo balanço na maratona. No dia a dia não, temos que focar em refatorar, extrair método, extrair código, extrair variável. Ainda mais em Java, que é muito fácil de fazer. Podemos fazer isso a rodo.

Se eu estivesse programando em uma IDE ou linguagem que não desse suporte à refatoração automática, ia bater uma certa preguiça. Não é a minha recomendação. É melhor usar uma IDE e uma linguagem que dão suporte à refatoração, e refatorar sempre o código, mesmo durante a maratona.

Vamos começar pelo TEST, aquele nosso primeiro problema. A primeira coisa a fazer é ler o enunciado.

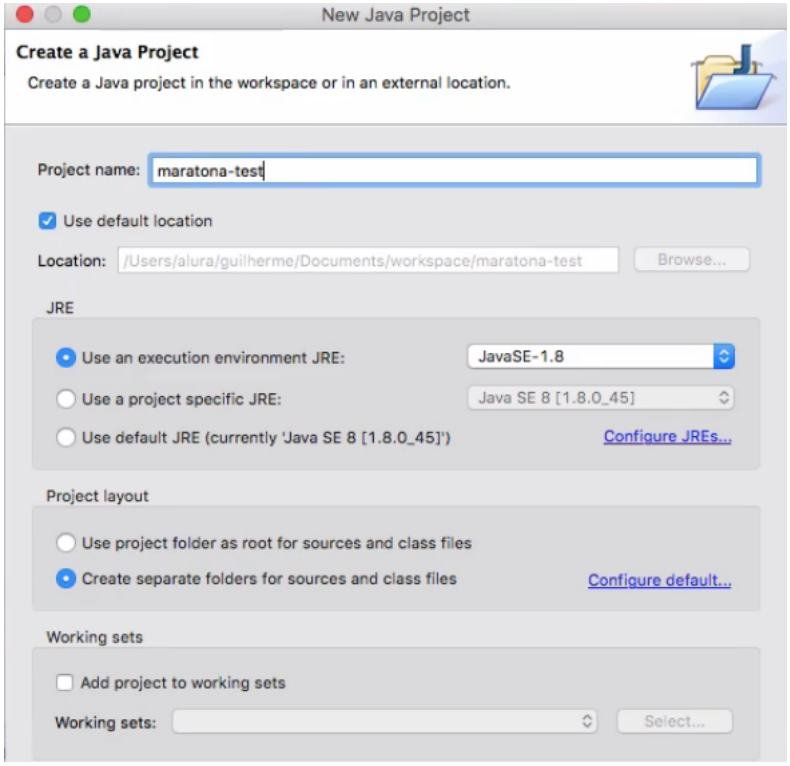
> Your program is to use the brute-force approach in order to find the Answer to Life, the Universe, and Everything. More precisely... rewrite small numbers from input to output. Stop processing input after reading in the number 42. All numbers at input are integers of one or two digits.

Devemos usar a força bruta para encontrar a reposta da vida, universo e tudo mais. Faremos isso reescrevendo números pequenos da entrada para a saída, parando ao encontrar o `42`. Todos os números são inteiros de um ou dois dígitos, ou seja, no máximo `100`. Deve ser entre `0` e `100`, eu imagino. Vamos ver a entrada e a saída.

```
Input:
1
2
88
42
99

Output:
1
2
88
```

Com isso mente, vamos criar um novo projeto no Eclipse que se chamará `maratona-test`, e a classe `Main`.



Com a classe criada, já podemos criar o método `main`.

```
public class Main {
  public static void main(String[] args){

  }
}
```

Já temos uma entrada também, que será a `entrada.txt`. Nela colaremos o input que o enunciado nos deu.

```
1
2
88
42
99
```

Voltamos ao problema. Começaremos com um `Scanner`.

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);

  }
}
```

Sabemos que precisamos ler linha a linha, e já conhecemos os perigos do `nextLine`. Usaremos então o `hasNext` como condição para o `while()`. Assim, lemos o próximo `int`, que chamaremos de `numero`. E se (`if`) ele for `42`, devemos parar (`break`). Do contrário, devemos imprimi-lo com um `sysout`.

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    while(sc.hasNext()) {
      int numero = sc.nextInt();
      if(numero == 42) break;
      System.out.println(numero);
    }

  }
}
```

Parece que é isso! Vamos rodar no terminal? Depois de entrar na pasta correta, temos:

```
Alura-Azul:bin alura$ java Main <entrada.txt
1
2
88
```

Aparentemente deu certo! Antes de enviar para o cliente, devemos testar algumas coisas a mais. Vou acrescentar alguns números na `entrada`, sabendo que o programa não deve imprimi-los.

```
1
2
88
42
99
2
2
2
34
42
```

Testando no terminal:

```
Alura-Azul:bin alura$ java Main <entrada.txt
1
2
88
Alura-Azul:bin alura$ java Main <entrada.txt
1
2
88
```

E se acrescentarmos mais um `42`?

```
1
42
2
```

```
88
42
99
2
2
2
34
42
```

Testando no terminal:

```
Alura-Azul:bin alura$ java Main <entrada.txt
1
2
88
Alura-Azul:bin alura$ java Main <entrada.txt
1
2
88
Alura-Azul:bin alura$ java Main <entrada.txt
1
```

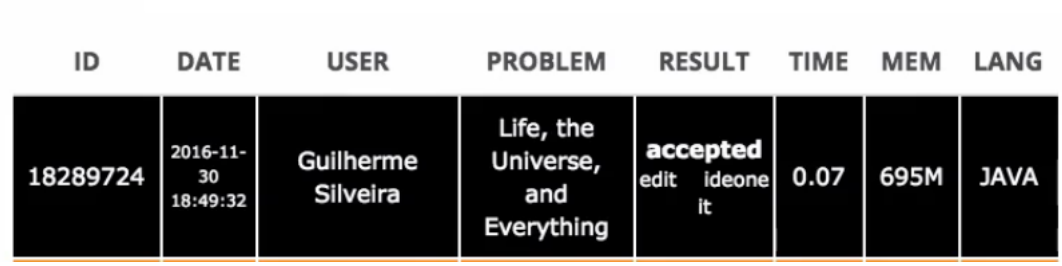
Beleza! E se ele vier ainda antes?

```
42
1
42
2
88
42
99
2
2
2
34
42
```

No terminal:

```
Alura-Azul:bin alura$ java Main <entrada.txt
1
2
88
Alura-Azul:bin alura$ java Main <entrada.txt
1
2
88
Alura-Azul:bin alura$ java Main <entrada.txt
1
Alura-Azul:bin alura$ java Main <entrada.txt
Alura-Azul:bin alura$
```

Parece ter dado certo mesmo. Vamos copiar o código e submetê-lo no Spoj, da maneira que já vimos.



ID	DATE	USER	PROBLEM	RESULT	TIME	MEM	LANG
18289724	2016-11-30 18:49:32	Guilherme Silveira	Life, the Universe, and Everything	**accepted** edit ideone it	0.07	695M	JAVA

E a nossa resolução foi aceita. O código já estava funcionando, passou e eu ganhei um ponto nessa prova da maratona. Até o próximo problema!