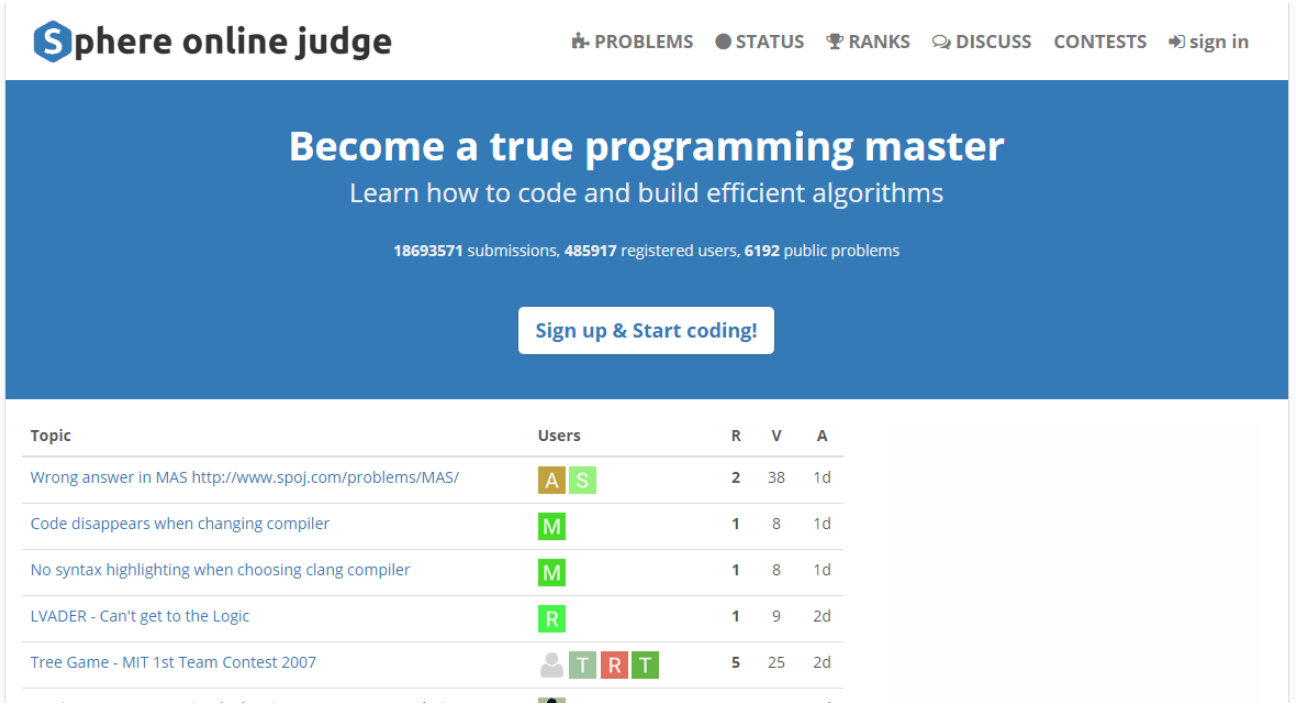
03

# Problema 42: Pensando e atacando o problema

### Transcrição

Vamos começar o primeiro problema. Ele está no site do Spoj, que é o Sphere Online Judge. O problema está nesse link (http://www.spoj.com/problems/TEST/).




É um problema bem básico, para que entendamos como funciona um juiz de código. Ele avaliará o comportamento do código, não a maneira que o código foi escrito. Vou focar, nessa aula e no curso, em mostrar que a maneira que pensamos durante a construção do código influencia no funcionamento do programa – inclusive se ele vai ou não funcionar. Criaremos alguns hábitos de programação saudáveis, que te ajudarão tanto nos programas da maratona quanto no dia a dia.

Vamos então para o problema em si. O seu enunciado é o seguinte:

> **TEST – life, the Universe, and Everything** 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.

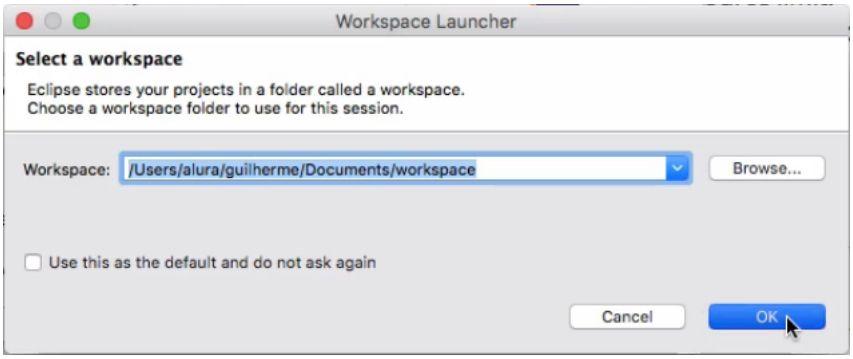
```
Example:

Input:
1
2
88
42
99

Output:
1
```

```
2
88
```

Vamos analisar juntos? O título é "Teste – A vida, o Universo e Tudo mais". Às vezes os problemas têm uma pegada engraçada, com alguma brincadeira, e às vezes eles se aproximam mais da vida real. O enunciado diz que o seu programa deverá encontrar a resposta da vida, do universo e tudo mais. Geralmente a introdução do problema tem algum segredo em que precisamos prestar atenção. Por enquanto não vou prestar tanta atenção, e deixarei algumas coisas passarem de propósito. O enunciado continua, dizendo que devemos reescrever números pequenos da entrada para a saída, e deu um exemplo.

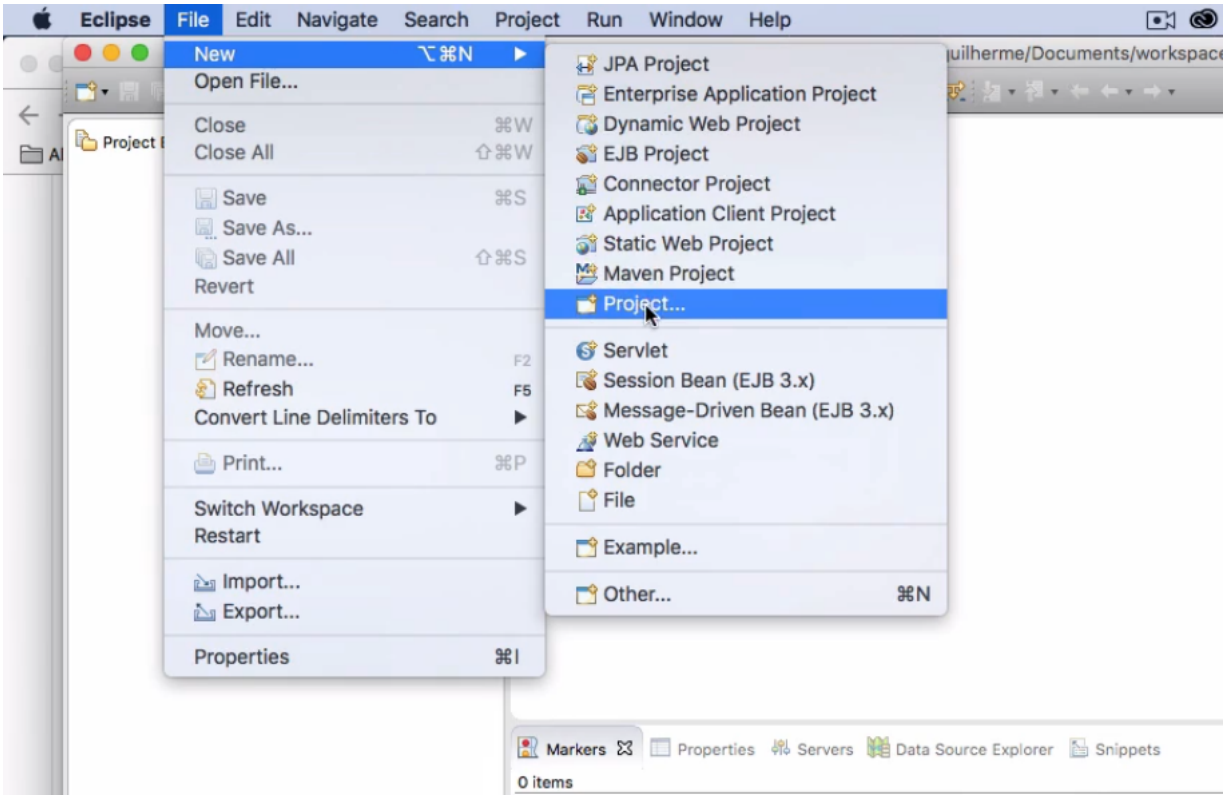
Repare que estou tomando cuidado para não falar tudo o que é relevante, para passarmos direto para o programa a ser feito e descubramos os erros que podem vir disso. Abriremos o Eclipse e faremos o código em Java. Lembrando que o foco do curso não é a linguagem de programação Java, mas a maneira de pensar e programar.

Se for a primeira vez que você abre o Eclipse, ele te pedirá o workspace. É bom que eu me lembra que defini o workspace como `Users/guilherme/alura/Documents/workspace` , para usar no terminal.

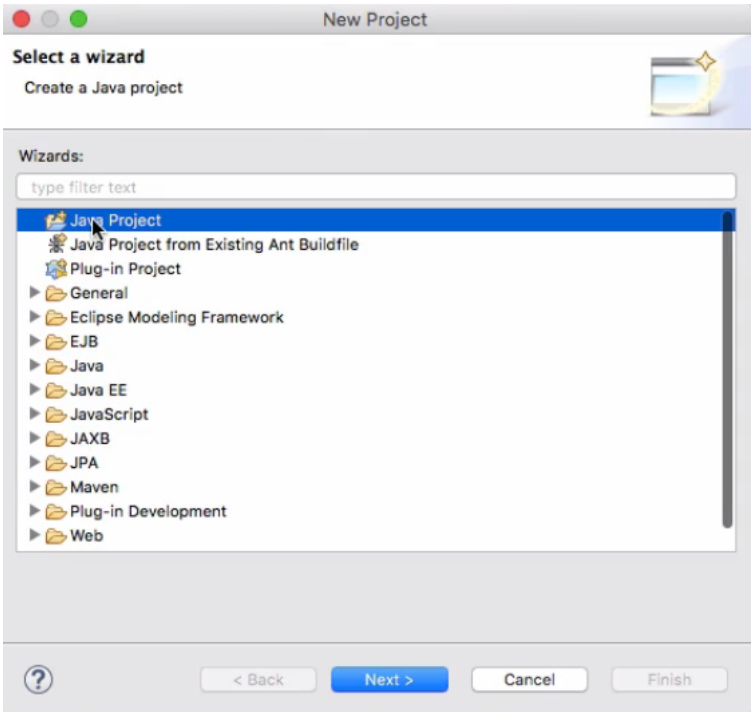



Assim que ele abre o programa, devemos clicar em `Workbench` .

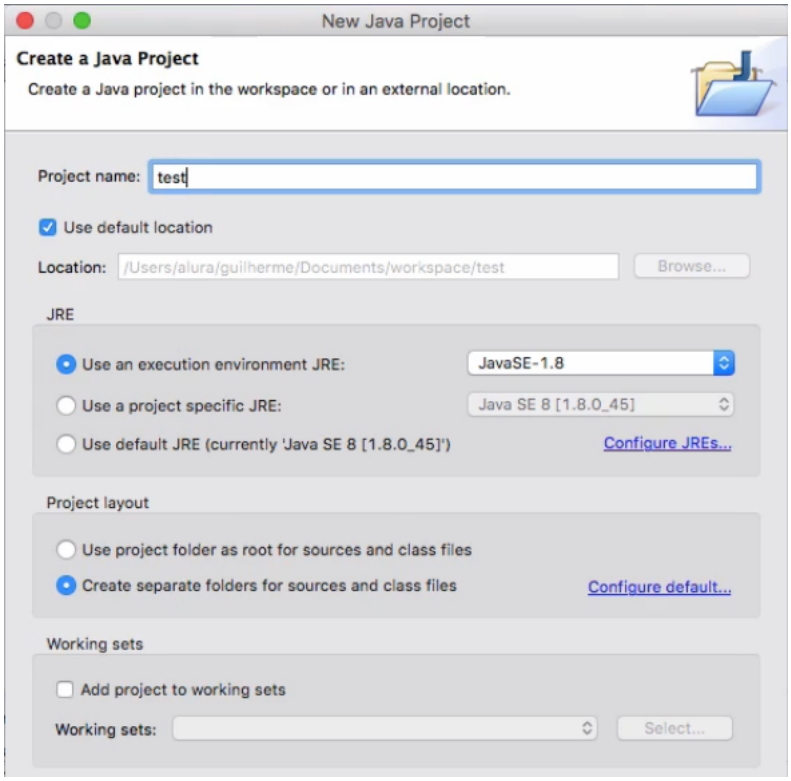

Então, criaremos um novo projeto, clicando em `File > New > Project` .




Informaremos que o projeto será feiro em Java clicando em `Java Project`.

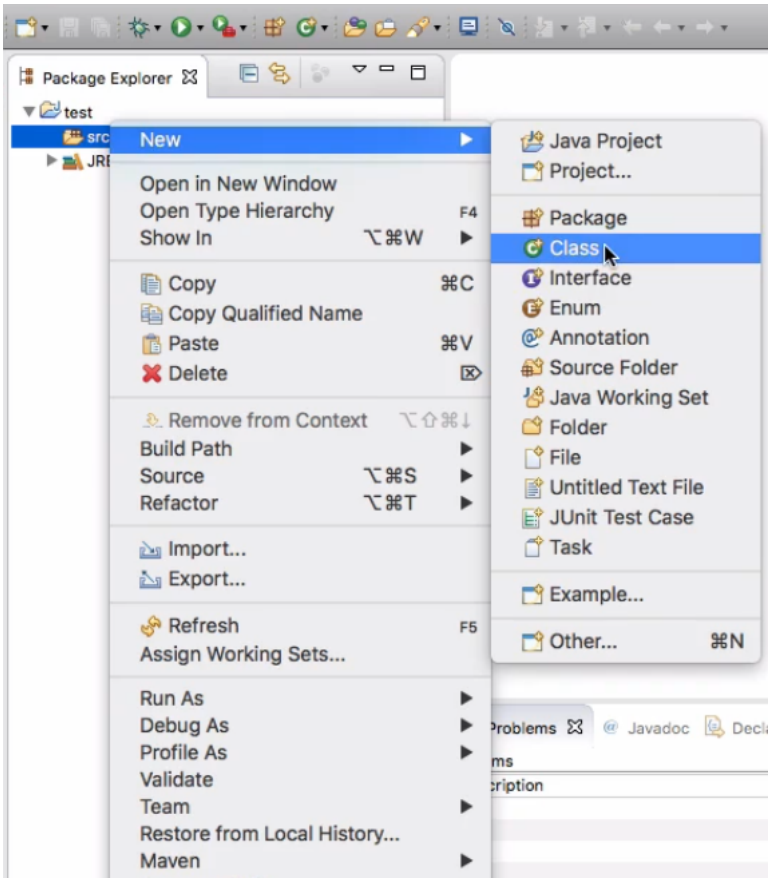



O nome do projeto será o nome do problema que estamos tentando resolver, já que o ideal é isolar cada problema em um projeto diferente. Assim, seu nome será `test`. Não por ser um teste nosso, mas porque o nome do problema é esse.




O Eclipse nos pergunta se queremos ir para uma perspectiva de Java, e aceitaremos.


Ele abre um projeto para nós, com o diretório `src`. Nele, criaremos uma nova classe. Para seguir o padrão, ela se chamará `Test`. Não que isso esteja necessariamente certo.



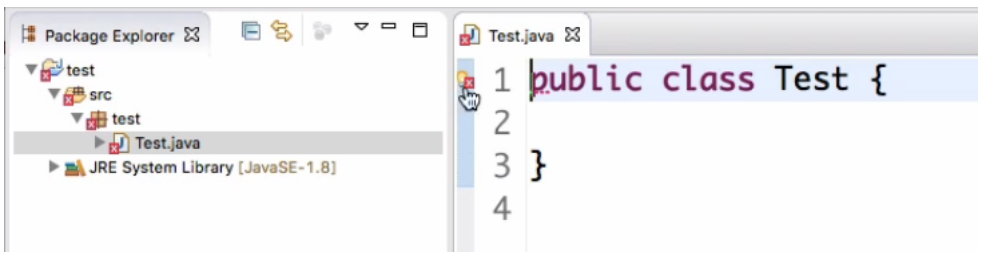

Teremos então o seguinte código:

```
package test;

public class Test {

}
```

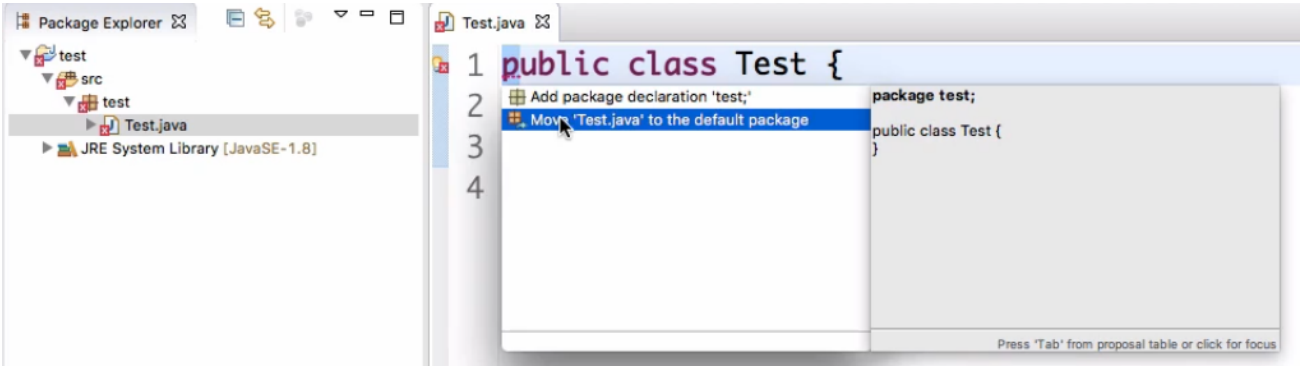
Ele criou a classe dentro de um pacote chamado `test`, e não era isso que queríamos. Removi essa linha, pois quero o pacote padrão.

```
public class Test {

}
```

Note os pequenos x's vermelhos. O programa está reclamando que estamos trabalhando no diretório errado, pois, se a classe não tem pacote, deve estar na raiz.




Podemos então mover para o diretório certo.




Agora, colocaremos o método `main` para começar o programa.

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

  }

}
```

Precisamos que o programa leia números da minha entrada. Podemos pedir para que ele leia uma linha inteira de uma vez só, e passe para a seguinte. Para que ele leia uma linha da nossa entrada padrão, que é o teclado, podemos usar diversos recursos do Java. A maneira mais simples é usar um `Scanner`. Ao criá-lo, precisamos dizer de onde ele tirará suas leituras. Nesse caso, será de `System.in`. Usaremos o `Command + Shift + O` para importar.

```
import java.util.Scanner;

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

  }

}
```

Então, podemos pedir para o `Scanner` nos dar a próxima linha, usando `nextLine()`, que é um número.

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    int linha = scanner.nextLine();

  }

}
```

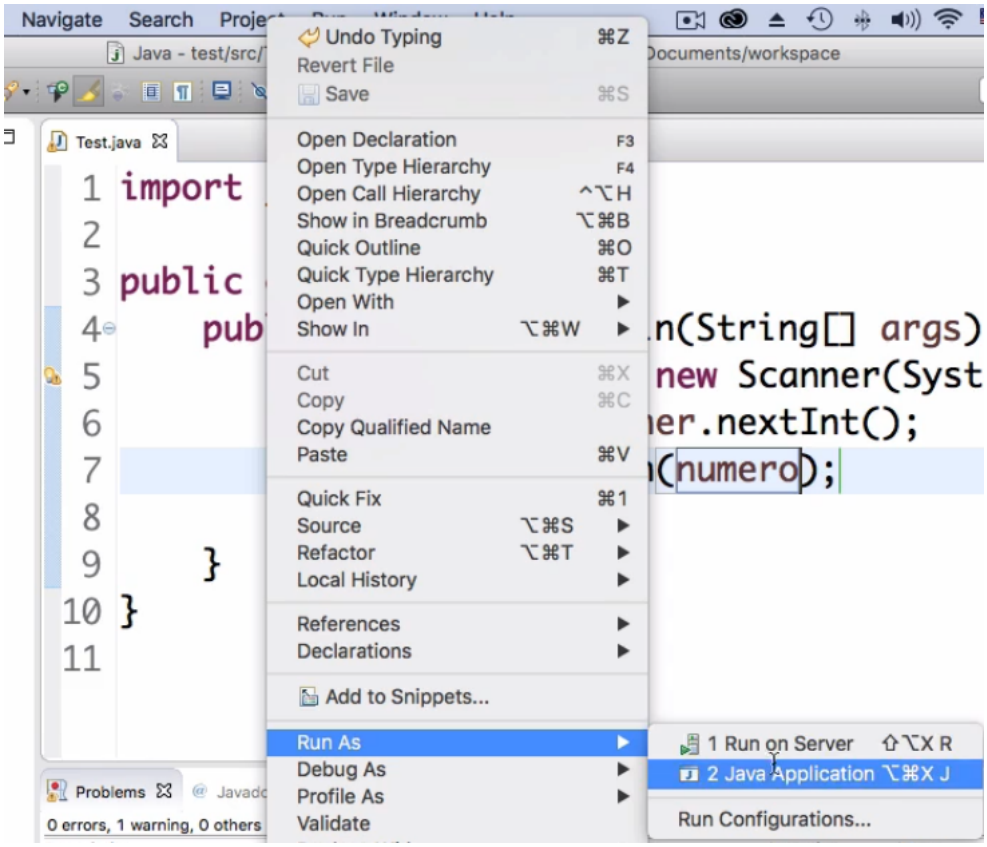
Entretanto, uma linha não é um `int`. É uma `String`. Portanto, temos que converter a string `1`, no número `1`. Seria um pouco ruim fazer isso toda vez. Felizmente, o `Scanner` é bonzinho, e tem o método `nextInt`. Então, se você sabe que o próximo trecho é um número inteiro, pode pedir para lê-lo com `nextInt`. Se houver dois números na mesma linha, separados com um espaço, ele pegará apenas o primeiro.

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    int numero = scanner.nextInt();

  }

}
```

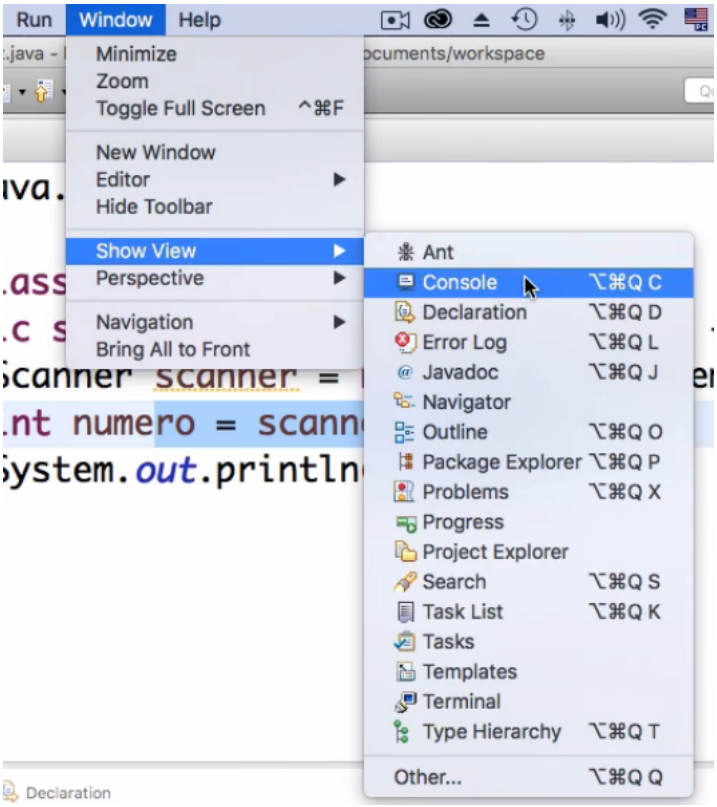
Agora que lemos o número, podemos imprimi-lo, usando `sysout` .

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    int numero = scanner.nextInt();
    System.out.println(numero);

  }

}
```

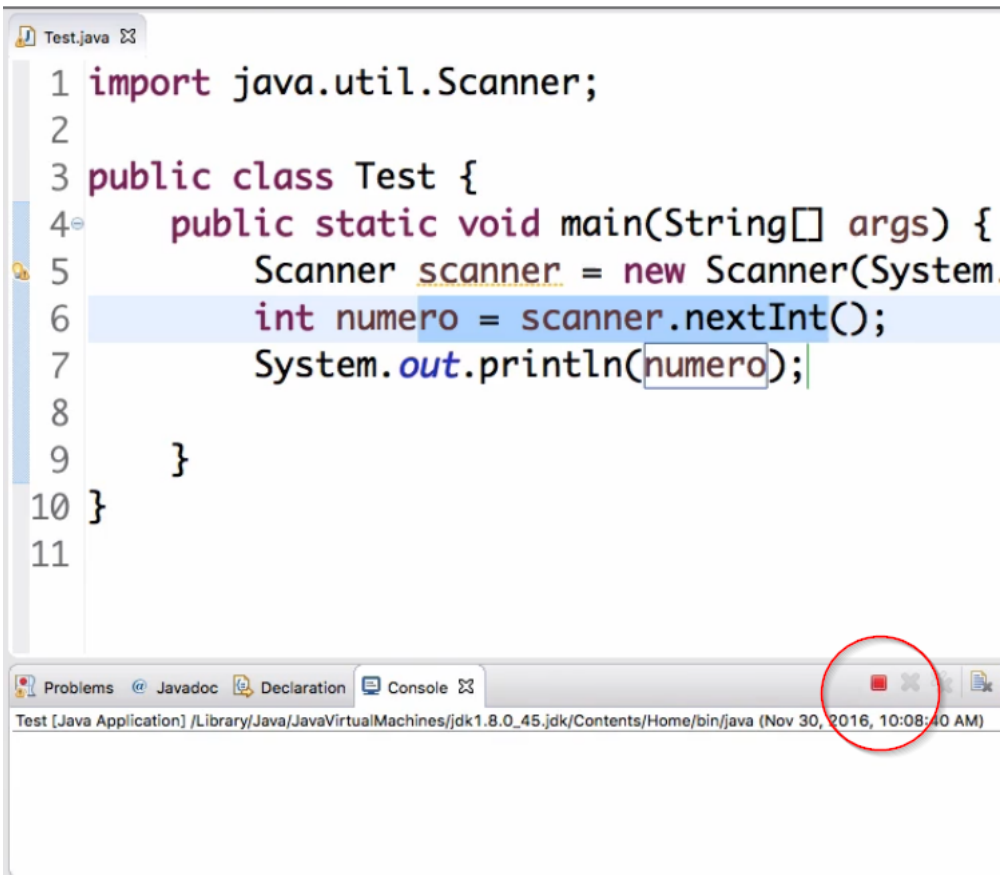
Vamos rodar esse programa, que está lendo apenas um número? Basta clicar com o botão direito e em `Run as > Java Application` , ou `Alt + Command + X + J` .




E o que acontece? Nada. O programa está esperando você digitar alguma coisa. Precisamos ir no console, onde há entrada e saída. Clicaremos em `Window > Show View > Console` .




O console ficara na parte inferior do programa, e possui um botão de Stop, para que pare de rodar, se necessário.



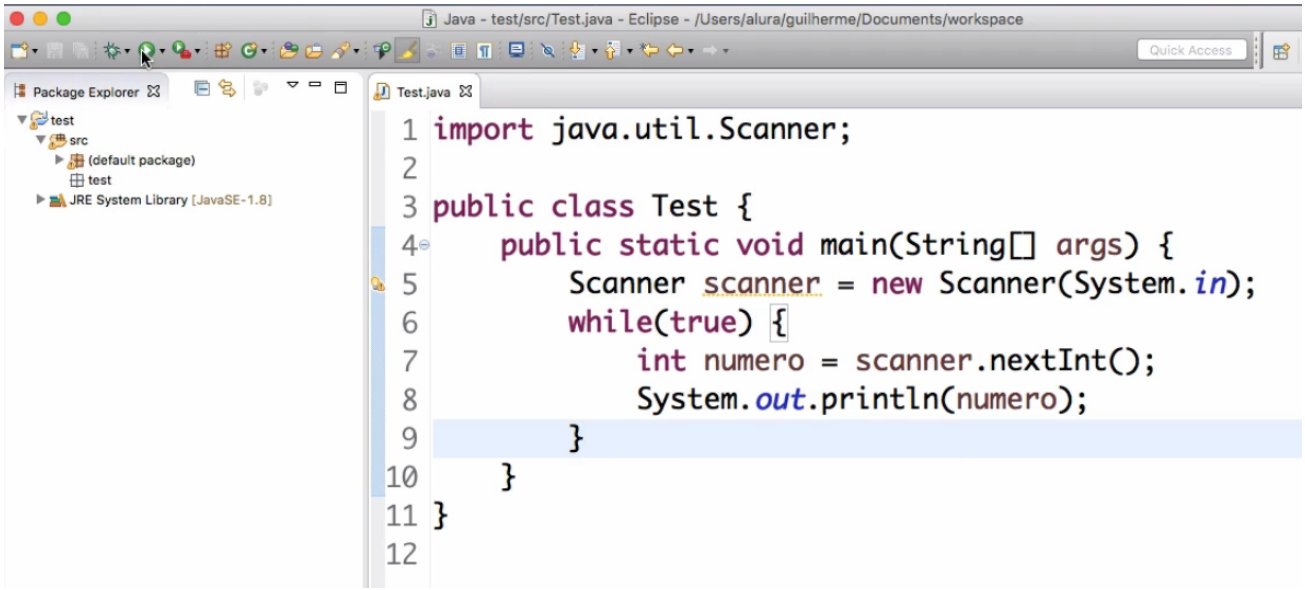

Nele, digitamos o número 56 e damos `Enter` . Em seguida, aparece novamente o número 56, que é o output.

```
56
56
```

E o programa para de rodar. O ideal é que ele continue a leitura por diversos número, sem parar. Para isso, precisamo de um laço, e o laço mais simples é o `while(true)` .

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    while(true) {
      int numero = scanner.nextInt();
      System.out.println(numero);

    }

  }

}
```

Usamos o que conhecemos da linguagem para que ele rode eternamente e imprima os números.




Ao rodar de novo, o programa espera que coloquemos um número no console. Colocaremos `1` .

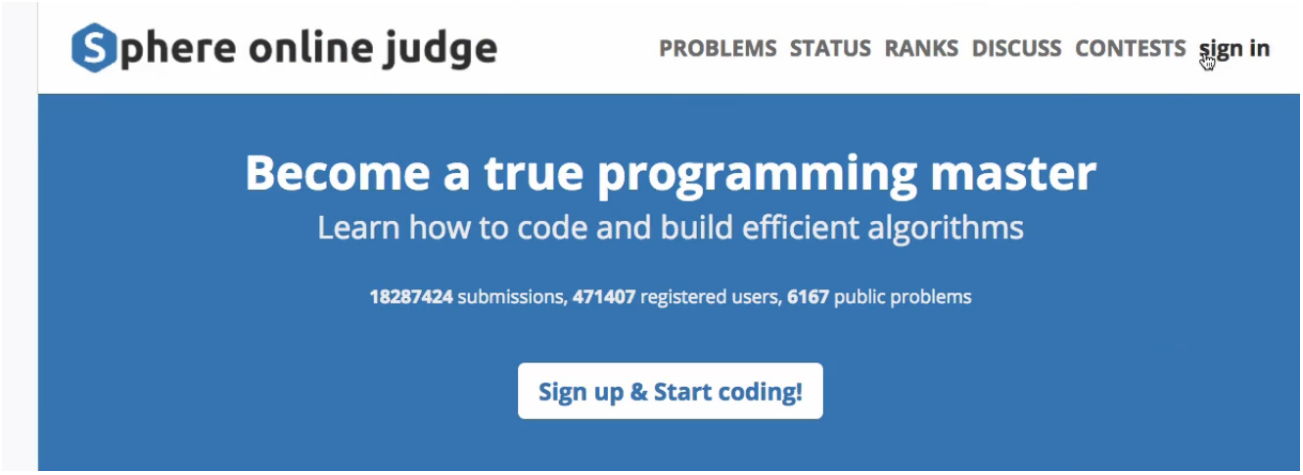
```
1
1
```

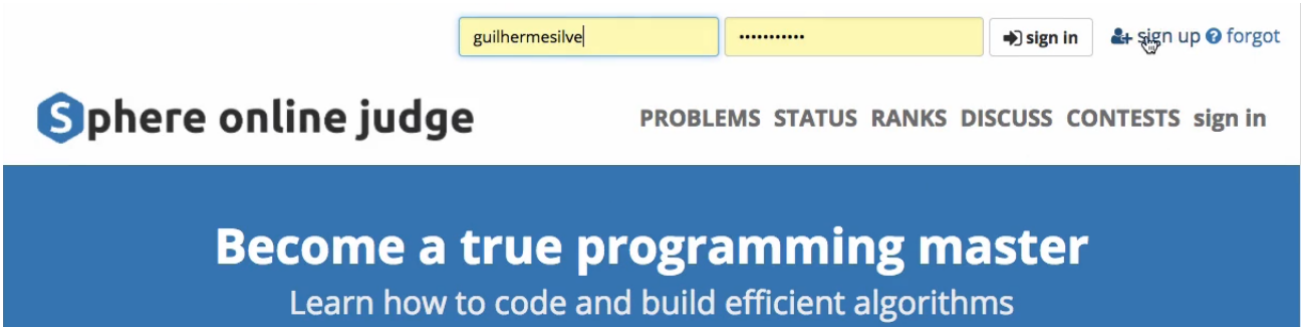
E ele prontamente nos devolve outro `1` . Testaremos colocar outro número, para ver se ele continua funcionando.

```
1
1
2
2
56
56
22
22
```

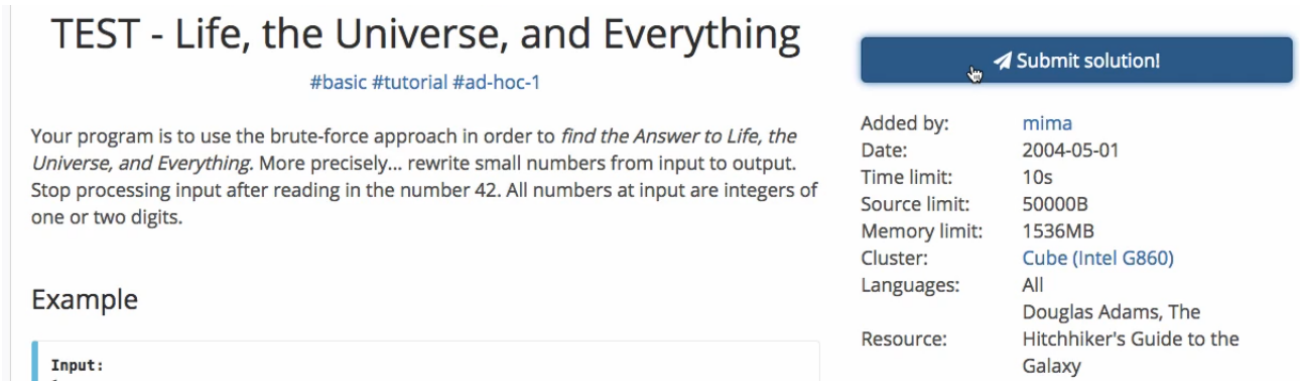
Ele respondeu a todos os números, então parece que tudo está funcionando direitinho. Vamos parar o programa e já enviar para o juiz. Repare que estamos nos precipitando ao escolher enviar sem atentar a todos os detalhes que mostrarei em breve.

Para enviar, nos logaremos no Spoj. Eu já tenho a minha conta.

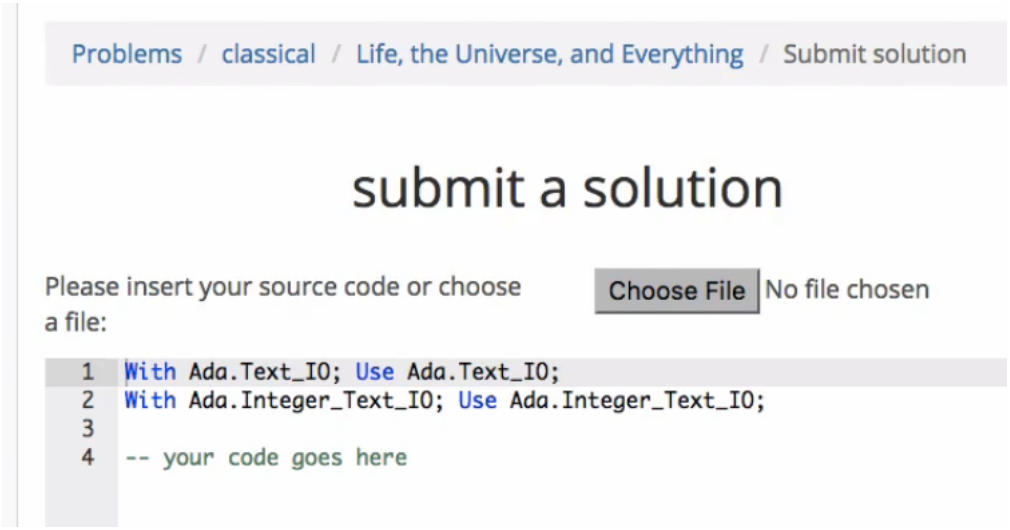


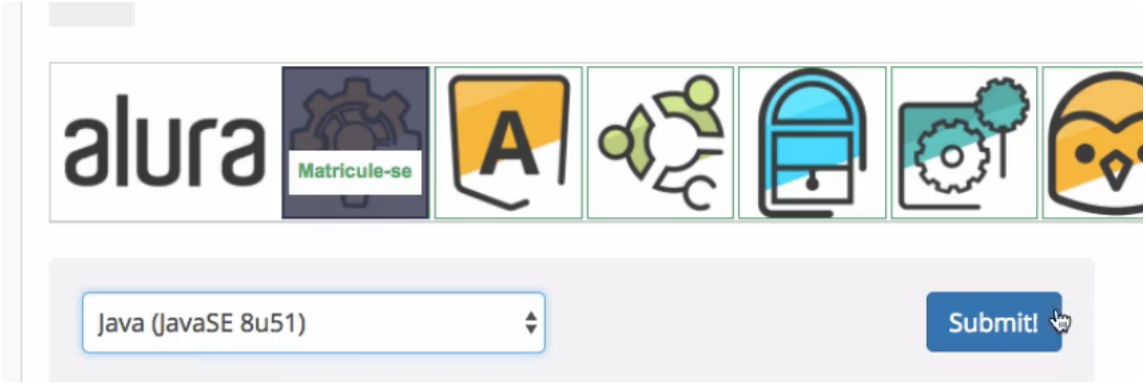

Caso você precise criar a sua, basta clicar em `Sign up`. Atualizaremos a tela do problema, para que ela perceba o nosso login e possamos submeter a resposta, clicando em `Submit solution`. Abriremos em uma aba nova com clicando com o `Command` pressionado.



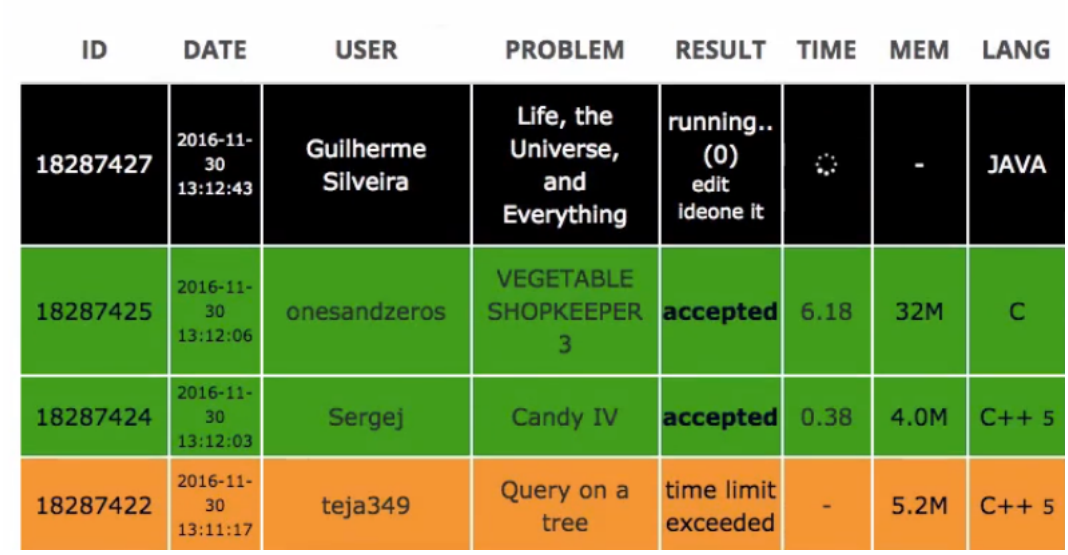

O Spoj permite que você envie um arquivo ou cole o código diretamente no campo.




Vamos optar por colar o código. Deletaremos tudo o que está ali e substituiremos pelo que acabamos de escrever. Abaixo desse campo, devemos selecionar a linguagem ( `Java (JavaSE 8u51)` ) e clicar em `Submit` .




O site nos levará para outra tela, que nos mostra que o nosso código está rodando.



| ID | DATE | USER | PROBLEM | RESULT | TIME | MEM | LANG |
|---|---|---|---|---|---|---|---|
| 18287427 | 2016-11-30 13:12:43 | Guilherme Silveira | Life, the Universe, and Everything | running.. (0) edit ideone it | | - | JAVA |
| 18287425 | 2016-11-30 13:12:06 | onesandzeros | VEGETABLE SHOPKEEPER 3 | **accepted** | 6.18 | 32M | C |
| 18287424 | 2016-11-30 13:12:03 | Sergej | Candy IV | **accepted** | 0.38 | 4.0M | C++ 5 |
| 18287422 | 2016-11-30 13:11:17 | teja349 | Query on a tree | time limit exceeded | - | 5.2M | C++ 5 |

E quando finalmente carrega, ele nos informa que a resposta está errada.

| ID | DATE | USER | PROBLEM | RESULT | TIME | MEM | LANG |
|---|---|---|---|---|---|---|---|
| 18287427 | 2016-11-30 13:12:43 | Guilherme Silveira | Life, the Universe, and Everything | wrong answer edit ideone it | 0.04 | 694M | JAVA |
| 18287425 | 2016-11-30 13:12:06 | onesandzeros | VEGETABLE SHOPKEEPER 3 | **accepted** | 6.18 | 32M | C |

O que pode estar errado? Qualquer coisa! Começaremos falando do laço, que vai rodando, independentemente do número inserido. Mas, em algum momento, o programa precisa parar. Vamos dar mais uma olhada no 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.

Até a parte em que o enunciado pede para jogar os números da entrada para a saída nós lemos com atenção. Mas então nos apressamos em fazer o código, e ignoramos algumas especificações. Logo em seguida ele pede para que paremos de processar depois do número 42. Repare que também no exemplo isso é explicitado.

```
Input:
1
2
88
42
99

Output:
1
2
88
```

Nós que fomos preguiçosos e lemos pela metade. Repare que erramos duas vezes na leitura: não escutamos o cliente até o fim, e olhamos o exemplo por cima.

Vamos rodar nosso programa novamente, emulando o caso que o cliente especificou. Digitaremos no console os números do exemplo, observando sua resposta:

```
1
1
2
2
88
42
42
```

Veja que o nosso programa continuou rodando depois que digitamos o `42`. Ou seja: erramos. Além disso, fica chato ficar digitando todas as vezes os mesmos números. O ideal é automatizar os testes, e é o que fazemos na prática. Em

uma maratona de programação, não temos muito tempo de ficar criando muitos testes de unidade. Então fazemos um arquivo separado, na pasta `src` exclusiva para testes. Clicaremos com o botão direito na pasta e em `New > File` .

| ID | DATE | USER | PROBLEM | RESULT | TIME | MEM | LANG |
|---|---|---|---|---|---|---|---|
| 18287427 | 2016-11-30 13:12:43 | Guilherme Silveira | Life, the Universe, and Everything | wrong answer edit ideone it | 0.04 | 694M | JAVA |
| 18287425 | 2016-11-30 13:12:06 | onesandzeros | VEGETABLE SHOPKEEPER 3 | **accepted** | 6.18 | 32M | C |

O novo arquivo se chamará `entrada1.txt` . Nele, colocaremos um teste muito simples, até mais simples que o exemplo dado pelo cliente do problema.

```
1
2
89
```

Depois, vamos ao **terminal** e entraremos no diretório que estamos usando. Sabemos que temos o diretório `test` , e que dentro dele o Eclipse criou o diretório `bin` , onde há o `text.class` no qual está a nossa `entrada1.txt` .

```
Alura-Azul:~ alura$ cd guilherme/Documents/workspace
Alura-Azul:workspace alura$ ls
RemoteSystemsTempFiles test
Alura-Azul:test alura$ ls
bin src
Alura-Azul:test bin alura$ cd bin
Alura-Azul:bin alura$ ls
Test.class  entrada1.txt test
```

Agora pediremos para que o programa leia de `entrada1.txt` .

```
Alura-Azul:~ alura$ cd guilherme/Documents/workspace
Alura-Azul:workspace alura$ ls
RemoteSystemsTempFiles test
Alura-Azul:test alura$ ls
bin src
Alura-Azul:test bin alura$ cd bin
Alura-Azul:bin alura$ ls
Test.class  entrada1.txt test
Alura-Azul: bin alura$ java Test < entrada1.txt>
```

E o resultado é o seguinte:

```
Alura-Azul:~ alura$ cd guilherme/Documents/workspace
Alura-Azul:workspace alura$ ls
RemoteSystemsTempFiles test
Alura-Azul:test alura$ ls
```

```
bin src
Alura-Azul:test bin alura$ cd bin
Alura-Azul:bin alura$ ls
Test.class  entrada1.txt test
Alura-Azul: bin alura$ java Test < entrada1.txt>
1
2
89
Exception in thread "main" java.util.NoSuchElementException
        at java.util.Scanner.throwFor(Scanner.java:862)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nextInt(Scanner.java:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at Test.main(Test.java:7)
```

Deu erro! E com um exemplo ainda mais simples que o do problema. Isso quer dizer que os testes no Eclipse nos geram uma situação irreal. Não adianta rodar apenas no Eclipse, se o cliente rodar apenas no terminal. No nosso caso, sabemos que o juiz roda no terminal, então temos que rodar como ele.

Qual foi o problema do nosso programa? Ele chegou ao final do arquivo e continuou lendo. Ele deveria parar ao final do arquivo. No nosso teste ele leu o `89` , imprimiu o `89` e procurou o próximo. Mas ele precisa parar se não houver mais nada para ler. Para isso, usaremos o `hasNext` . Se ( `if` ) não houver mais nada, o programa deve para ( `break` ).

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    while(true) {
      int numero = scanner.nextInt();
      System.out.println(numero);
      if(!scanner.hasNext()) break;

    }

  }

}
```

Salvaremos e testaremos no terminal, depois de limpar sua tela.

```
Alura-Azul: bin alura$ java Test < entrada1.txt
1
2
89
Alura-Azul:bin alura$
```

Agora funcionou! Para esse exemplo, ao menos, que é um bom começo por ser mais simples do que o proposto no problema. Vamos testar com esse agora? Não podemos deixar que o cliente teste exatamente o que ele pediu e encarar um erro. Criaremos a `entrada2.txt` , da mesma forma que criamos a primeira. Nela, constará:

```
1
2
88
42
99
```

Agora, rodaremos esse segundo teste no terminal.

```
Alura-Azul: bin alura$ java Test < entrada1.txt
1
2
89
Alura-Azul:bin alura$ java Test < entrada2.txt
1
2
88
42
99
```

Não deu tão certo. Era para ele imprimir até o `88`, pois deve parar no `42`. Para corrigir isso, precisamos voltar ao código. Se ( `if` ) o número for `42`, quero que o programa pare ( `break` ).

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    while(true) {
      int numero = scanner.nextInt();
      if(numero == 42) break;

      System.out.println(numero);
      if(!scanner.hasNext()) break;

    }

  }

}
```

Repare que estamos cobrindo os dois `if` s com dois testes: o `entrada1.txt` cobre o caso em que não há mais o que ler, e o `entrada2.txt` cobre o caso do `42`.

Vamos testar novamente no terminal? Primeiro testaremos a `entrada2.text`

```
Alura-Azul: bin $alura java Test < entrada2.txt
1
2
88
```

Funcionou! Será que o primeiro exemplo ainda funciona?

```
Alura-Azul: bin alura$ java Test < entrada1.txt
1
2
89
```

Podemos tentar enviar esse código para o juiz. Copiaremos tudo e colaremos naquele campo de submissão. Depois de um tempo compilando, temos uma surpresa:

| ID | DATE | USER | PROBLEM | RESULT | TIME | MEM | LANG |
|---|---|---|---|---|---|---|---|
| 18287451 | 2016-11-30 13:18:49 | Guilherme Silveira | Life, the Universe, and Everything | compilation error edit ideone it | - | - | JAVA |
| 18287450 | 2016-11-30 13:18:48 | IndraDeza | Life, the Universe, and Everything | compilation error | - | - | C++ 4.3.2 |

Um erro de compilação! Vamos fechar essa tela e abrir novamente a tela de submissão. Quando olhamos com atenção, vemos que havia um o código antes de colarmos o nosso por cima. Ele é o seguinte:

```
import java.util.*;
import java.lang.*;

class Main
{
    public static void main (String[] args) throws java.lang.Exception
    {

    }
}
```

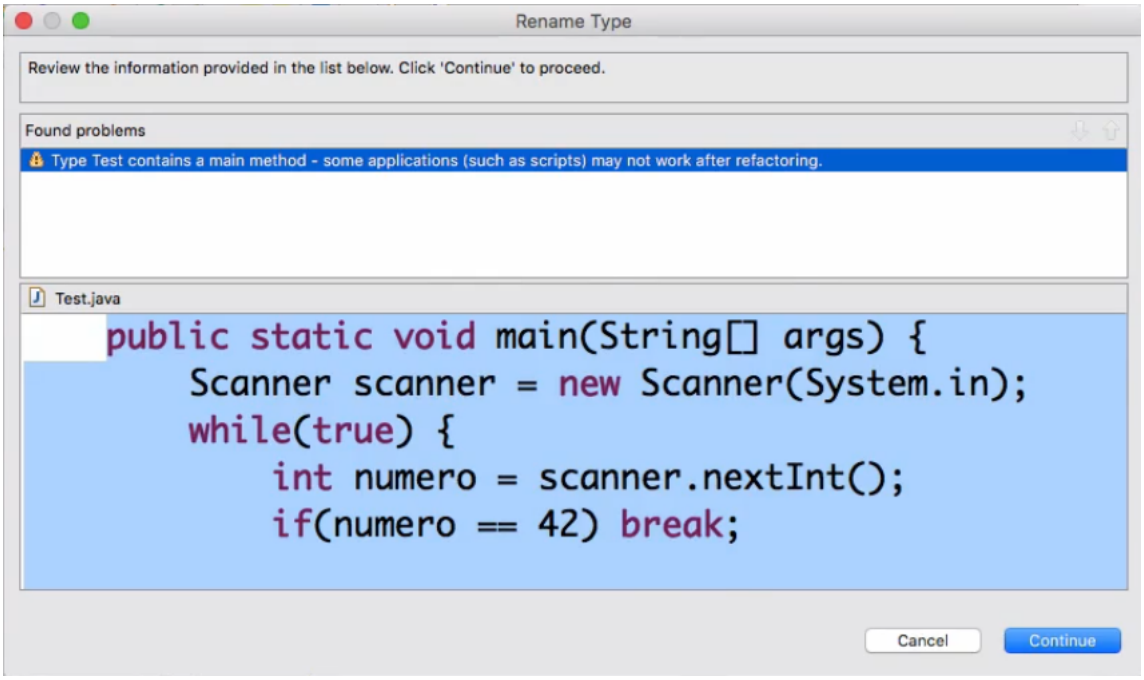
Repare no nome da classe que temos que enviar: `Main`. Não é `Test`, como havíamos presumido. Novamente, o programador não leu a especificação do cliente. Ele pediu que fosse no pacote padrão, pois não está escrito nada de `package`. Mas a classe deveria ser `Main`.

Assim, vamos renomear a classe no nosso código.

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    while(true) {
      int numero = scanner.nextInt();
      if(numero == 42) break;

      System.out.println(numero);
      if(!scanner.hasNext()) break;

    }
```

```
    }

}
```

O programa vai pedir para renomear os arquivos, e vamos aceitar.




Agora sim, podemos copiar tudo e colar sobre o código que lá estava. Quando submetemos, veremos o site nos responder com o seguinte:

| ID | DATE | USER | PROBLEM | RESULT | TIME | MEM | LANG |
|---|---|---|---|---|---|---|---|
| 18287466 | 2016-11-30 13:20:41 | Guilherme Silveira | Life, the Universe, and Everything | **accepted** edit ideone it | 0.06 | 695M | JAVA |
| 18287465 | 2016-11-30 | vikram | Buying Apples! | wrong answer | 0.07 | 695M | JAVA |

O código foi aceito! Conseguimos fazer de acordo com o que o cliente pediu. Até a próxima!