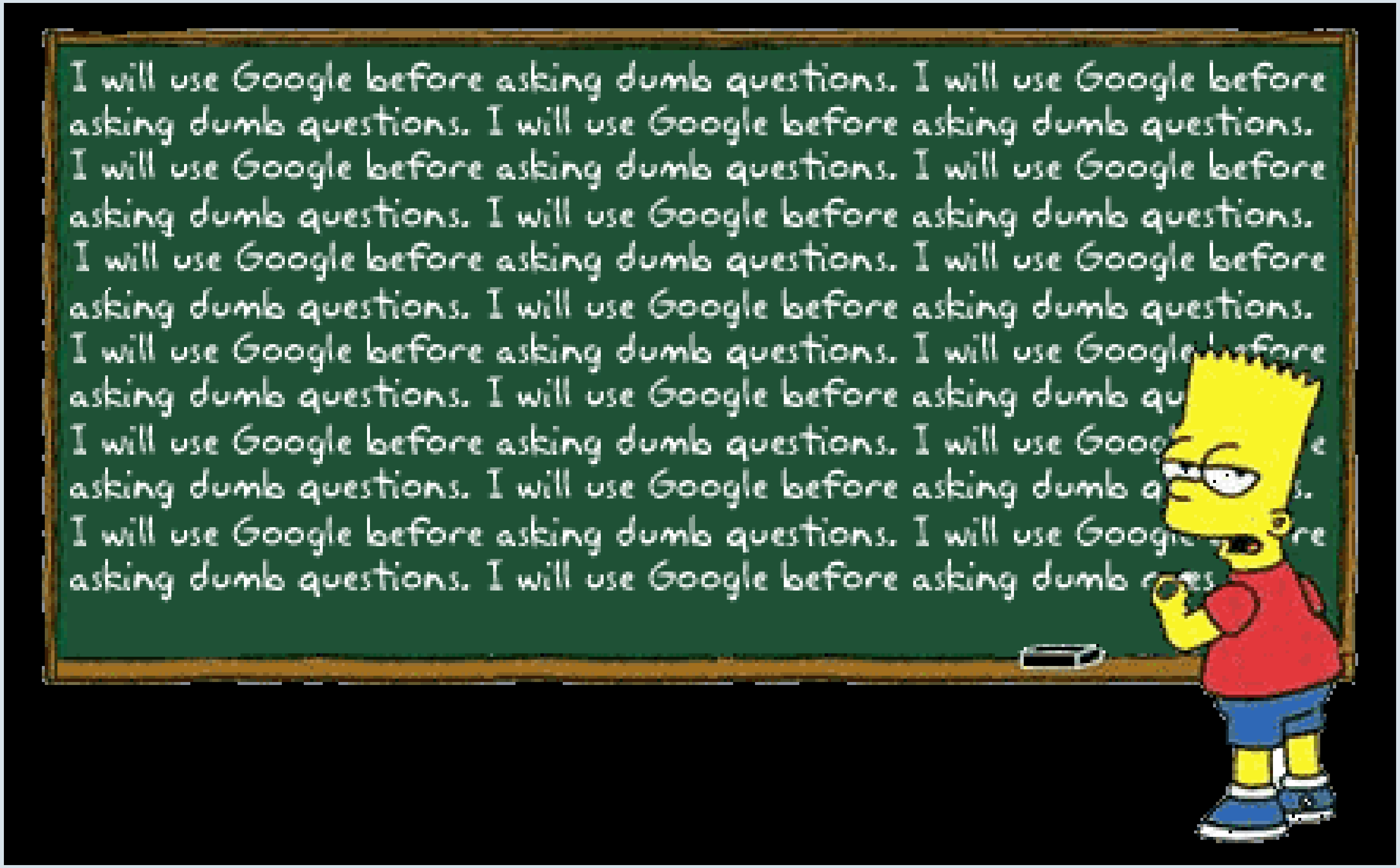
Repetição de Comandos

Fernanda Barbosa
Fernando Birra
Luís Caires
Armanda Rodrigues

O Ecoador

O Bart está de castigo



O Ecoador

- Um ecoador é um objecto da classe `Echo` , que possui apenas duas operações (`echo e countEchoes`).
- A operação `echo` recebe uma string e um número inteiro positivo `k`, e devolve a concatenação de `k` cópias da string dada.
- A operação countEchoes devolve o total de ecos “ecoados”
- Por exemplo:

```
Echo e = new Echo();
e.echo(“p-p-poker face!”,2)
“p-p-poker face!p-p-poker face!” (String)
e.echo(“mum ”,4)
“mum mum mum mum ” (String)
e.countEchoes()
6 (int)
```

Programando a classe Echo

```
public class Echo {
  private int echoCounter;
  public Echo() { echoCounter = 0; }
  public String echo(String data,int rep) {
      int count; // contador de copias já repetidas
      String copies; // string com cópias já repetidas
      count = 0;
      copies = “”;
      while (count != rep) {
        copies = copies+ data;
        count = count + 1;
      }
      echoCounter = echoCounter+rep;
      return copies;
   }
   public int countEchoes { return echoCounter; }
}
```

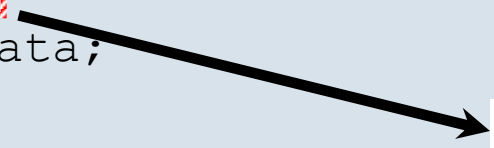
Programando a classe Echo

```
public class Echo {
  private int echoCounter;
  public Echo() { echoCounter = 0; }
  public String echo(String data,int rep) {
      int count; // contador de copias já repetidas
      String copies; // string com cópias já repetidas
      count = 0;
      copies = “”;
      while (count != rep) {
        copies = copies+ data;
        count = count + 1;
      }
      echoCounter = echoCounter+rep;
      return copies;
   }
   public int countEchoes { return echoCounter; }
}
```

Ciclo “while”

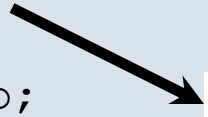
Programando a classe Echo

```
public class Echo {
  private int echoCounter;
  public Echo() { echoCounter = 0; }
  public String echo(String data,int rep) {
      int count; // contador de copias já repetidas
      String copies; // string com cópias já repetidas
      count = 0;
      copies = "";
      while (count != rep) {
        copies = copies+ data;
        count = count + 1;
      }
      echoCounter = echoCounter+rep;
      return copies;
   }
   public int countEchoes { return echoCounter; }
}
```



Programando a classe `Echo`

```
public class Echo {
  private int echoCounter;
  public Echo() { echoCounter = 0; }
  public String echo(String data,int rep) {
      int count; // contador de copias já repetidas
      String copies; // string com cópias já repetidas
      count = 0;
      copies = "";
      while (count != rep) {
        copies = copies+ data;
        count = count + 1;
      }
      echoCounter = echoCounter+rep;
      return copies;
   }
   public int countEchoes { return echoCounter; }
}
```



Corpo do ciclo

O Ciclo while

- Enquanto a condição do ciclo for verdadeira, os comandos no corpo do ciclo serão executados repetidamente
- Se inicialmente a condição for falsa, o corpo do ciclo nunca chegará a ser executado ...
- Caso contrário, será executado uma vez, sendo depois a condição avaliada de novo ...
- Se continuar a ser verdadeira, o corpo será executado de novo, e a condição reavaliada ...
- O ciclo só pára de repetir quando a condição for falsa (o que pode nunca acontecer, oops!)

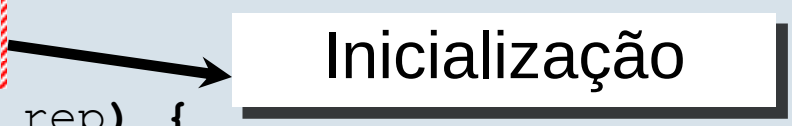
O Ciclo while

- Os três elementos fundamentais de um ciclo são
  - Inicialização
    - Estabelece os valores iniciais das variáveis que podem ser alteradas em cada passo do ciclo.
  - Condição
    - Indica se o ciclo deve repetir mais um passo.
  - Corpo
    - Indica o que deve ser feito em cada passo do ciclo
    - Um “passo” também se chama “iteração”

```
Inicialização ;
while ( Condição )
    {
      Corpo;
    }
```

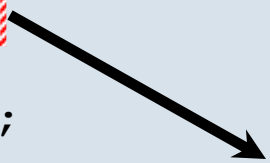
Programando a classe `Echo`

```
public class Echo {
  private int echoCounter;
  public Echo() { echoCounter = 0; }
  public String echo(String data,int rep) {
      int count; // contador de copias já repetidas
      String copies; // string com cópias já repetidas
      count = 0;
      copies = “”;
      while (count != rep) {
        copies = copies+ data;
        count = count + 1;
      }
      echoCounter = echoCounter+rep;
      return copies;
   }
   public int countEchoes { return echoCounter; }
}
```



Programando a classe `Echo`

```
public class Echo {
  private int echoCounter;
  public Echo() { echoCounter = 0; }
  public String echo(String data,int rep) {
      int count; // contador de copias já repetidas
      String copies; // string com cópias já repetidas
      count = 0;
      copies = “”;
      while (count != rep) {
        copies = copies+ data;
        count = count + 1;
      }
      echoCounter = echoCounter+rep;
      return copies;
   }
   public int countEchoes { return e
}
```

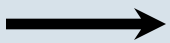


Corpo do Ciclo

Altera o valor da variável `count`

Programando a classe `Echo`

```
public class Echo {
  private int echoCounter;
  public Echo() { echoCounter = 0; }
  public String echo(String data,int rep) {
      int count; // contador de copias já repetidas
      String copies; // string com cópias já repetidas
      count = 0;
      copies = “”;
      while (count != rep) {
        copies = copies+ data;
        count = count + 1;
      }
      echoCounter = echoCounter+rep;
      return copies;
   }
   public int countEchoes { return echoCounter; }
}
```



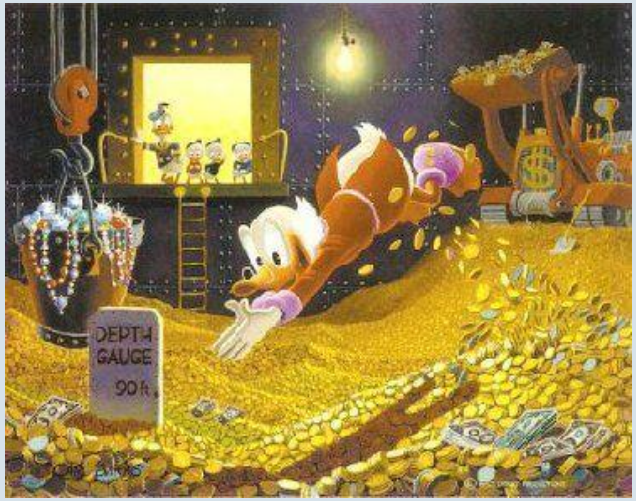
Condição do Ciclo

Quando o valor da variável `count` for igual ao de `rep`, o ciclo termina

Poupanças

Poupanças

- Recupere a sua classe `SafeBankAccount`, cujos objectos representam contas bancárias. Vamos acrescentar uma nova operação a essa classe.
- Programe a sua classe no Eclipse.
- Teste um (ou vários) objectos `SafeBankAccount`, e verifique que se comportam como se espera.



Recorde a Conta Bancária Segura

- Operações (interface de SafeBankAccount):

```
public void deposit(int amount)
```

Depositar a importância `amount` na conta

```
public void withdraw(int amount)
```

Levantar a importância `amount` na conta ou aplicar a multa

```
public int getBalance()
```

Consultar o saldo da conta

```
public boolean redZone()
```

Indica se a conta está devedora

```
public int computeInterest()
```

Calcular qual o valor do juro anual a aplicar

```
public void applyInterest()
```

Creditar o juro anual ao saldo da conta

Exemplo de poupança

- Imagine uma conta com um saldo inicial de 10000 e uma taxa de juro anual de 5%
- Recorde que todos os valores são apresentadoe em cêntimos. Se necessário, faça os arredondamentos correspondentes.

Ano	Saldo
0	10000
1	10500
2	11025
3	11576
4	...

Vamos lá poupar durante uns anos...

- A classe SafeBankAccount já tinha forma de fixar uma taxa de juro, calcular o juro anual e creditar esse juro numa conta
- Acrescente a função `howManySavingsYears`

```
public int howManySavingsYears(int target)
```

Calcula o número de anos necessários para que o saldo da conta seja maior ou igual ao objectivo. Devolve -1, caso não seja possível atingir o saldo dado (conta devedora).

- Teste a sua classe num programa interactivo.

Nobody Expects the Spanish Inquisition!

A Inquisição Espanhola

- Defina em Java uma classe `Cardinal` cujos objectos são os personagens de um famoso sketch dos Monty Python, em que vários cardeais da Inquisição tentam arrancar confissões com torturas peculiares.
- Programe a sua classe no Eclipse.
- Teste objectos `Cardinal`, e verifique que se comportam como se espera.

A Inquisição Espanhola

- Cada inquisidor caracteriza-se por um nome e um grau de perícia com cada instrumento de tortura. Este grau de perícia corresponde aos "pontos de tortura" que o inquisidor aplica, quando usa cada instrumento. Os instrumentos de tortura à disposição são:
  - Uma grelha de escorrer loiça lavada
  - Uma almofada fofinha
  - Uma poltrona confortável
- Os inquisidores aplicam sucessivas torturas, escolhidas de forma aleatória, às suas vítimas, até obter uma confissão. Todas as vítimas confessam, quando o seu número de "pontos de resistência" se esgota, após sucessivas aplicações de "pontos de tortura".

A Inquisição Espanhola

- Operações reconhecidas (Interface de `Cardinal`):

  `public int` `tieRack()`
  devolve os “pontos de tortura” resultantes de atar uma grelha de escorrer loiça lavada à vítima

  `public int` `pokeWithSoftCushion()`
  devolve os “pontos de tortura” resultantes de bater na vítima com uma almofada fofinha

  `public int` `seatOnTheComfyChair()`
  devolve os “pontos de tortura” resultantes de obrigar a vítima a sentar-se numa poltrona muito confortável

A Inquisição Espanhola

- Operações reconhecidas (interface de `Cardinal`)

  `public` `String getLastVictim()`

  devolve o nome da última vítima

  `public` `String getLastDescription()`

  Devolve uma descrição detalhada da tortura (uma String consistindo nos instrumentos de tortura, separados por vírgulas

  `void` `torture(String victim,` **`int`** `resistance)`

  Tortura uma vítima até lhe arrancar a confissão, quando os pontos de tortura acumulados superarem os pontos de resistência da vítima, guardando uma String a explicar detalhadamente a tortura.

A Inquisição Espanhola

- Este problema é inspirado na "barra de energia" dos personagens de jogos de computador. Nos jogos de combate, quando a energia se esgota, a personagem é derrotada. Diferentes ataques roubam diferentes quantidades de energia à personagem. Aqui, o princípio é o mesmo.
- À partida, não sabemos quantas vezes o ciclo se vai executar. Sabemos que ele se executa tantas quantas as necessárias para a resistência do torturado ficar menor ou igual a zero.
  - Repare que se o torturado tiver resistência 0 confessa logo!
  - **Um ciclo while executa-se 0 ou mais vezes.**
- Ao construir objectos da classe `Cardinal`, tenha o cuidado de definir os "pontos de tortura" como inteiros positivos, caso contrário, o ciclo pode nunca acabar!

A Inquisição Espanhola

- Use uma constante para cada tipo de tortura (por exemplo, 0 para grelha, 1 para almofada, 2 para cadeirão)
  - Para decidir qual o golpe seguinte, sorteie um número inteiro aleatório entre 0 e 2. Aplique o golpe, removendo à vítima os pontos de resistência correspondentes.
  - Para calcular um número aleatório entre 0 e 2, use a classe `Random`, da biblioteca `java.util`. Não se esqueça de a importar:

```
import java.util.random;
...
   void torture(...) {
        int nextTorture;
        Random r;
        ...
        nextTorture = r.nextInt(3); /* devolve um inteiro entre 0 e 3 */
        ...
   }
...
```

A Inquisição Espanhola

```
/* Excerto de exemplo do programa principal (note que o seu programa principal deve ser
   interactivo, e este exemplo não é), para ilustrar a interface da classe. */
Cardinal biggles = new Cardinal("Biggles", 2, 4, 3);
Cardinal fang = new Cardinal("Fang", 1, 400, 3000);
System.out.println(biggles.tieRack());
System.out.println(biggles.pokeWithSoftCushion());
System.out.println(biggles.seatOnTheComfyChair());
biggles.torture("Old Lady", 10);
System.out.println(biggles.getLastDescription());
System.out.println(biggles.getLastVictim());
fang.torture(“Chuck Norris”, 5000);
System.out.println(fang.getLastVictim() + fang.getLastDescription());
```

```
2
4
3
soft cushion, comfy chair, rack, rack,
Chuck Norris soft cushion, comfy chair, rack, comfy chair,
```

A Inquisição Espanhola

- Excerto de exemplo de execução na consola do Eclipse:

```
Cardinal name:
Biggles
Rack skill:
2
Cushion skill:
3
Chair skill:
4
How many victims shall Cardinal Biggles torture today?
2
Who shall Cardinal Biggles torture today?
Miss Daisy
How many resistance points?
20
Cardinal Biggles tortured Miss Daisy with a sequence of cushion, comfy chair,
cushion, rack, comfy chair, cushion, rack, after which the tortured victim
confessed everything.
Who shall Cardinal Biggles torture today?
Chuck Norris
...
```