IP – Semana 3 – Lista de Problemas (alunos de 1ª inscrição)

Miguel Goulão

miguel.goulao@di.fct.unl.pt

Objectivos

- No final da 3ª semana, o aluno deverá ser capaz de:
  - Usar o BlueJ na construção de classes simples, definindo:
    - Constantes (públicas)
    - Variáveis de instância (privadas)
      - Usando tipos elementares como boolean, int, float e double
    - Múltiplos construtores de instâncias da classe (públicos)
    - Operações modificadoras e de consulta (públicas)
    - Listas de parâmetros de operações
    - Chamadas a operações da classe dentro de outras operações
    - Operações aritméticas na implementação de métodos
      - Usando, quando necessário, operações da biblioteca Math
    - Operações lógicas na implementação dos métodos
    - Variáveis locais/temporárias dentro dos métodos
    - Estruturas de controle dentro das operações

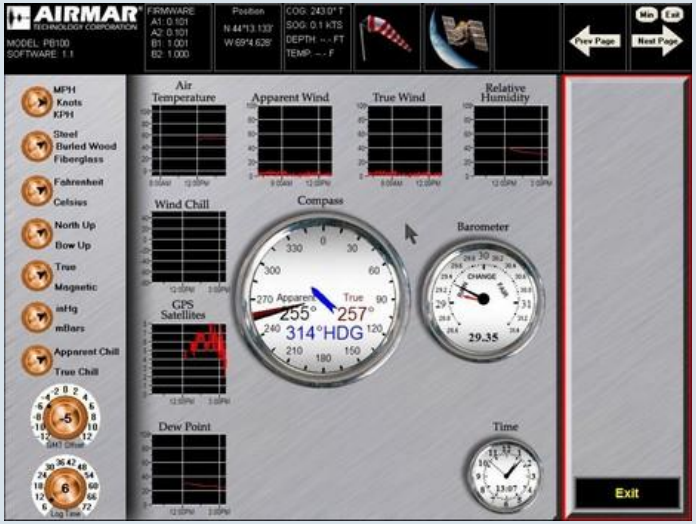
Especificação de classes

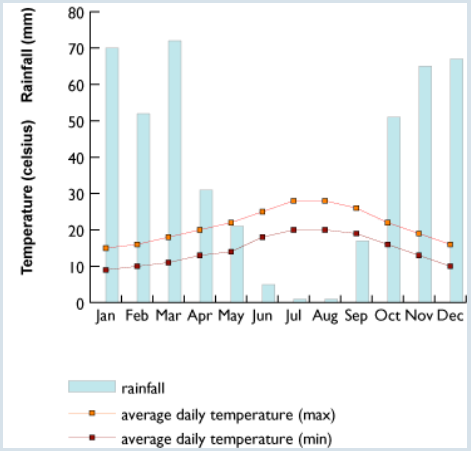
ESTAÇÃO METEOROLÓGICA
JOGO DA ADIVINHA
ROBIN HOOD

Estação Meteorológica

Estação Meteorológica

- Defina em Java uma classe `WeatherStation` cujos objectos representam uma estação meteorológica.
- Programe a sua classe no BlueJ.
- Teste um (ou vários) objectos `WeatherStation`, e verifique se se comportam como se esperada.





Estação Meteorológica

- Cada objecto `WeatherStation` recebe valores de temperatura ao longo do tempo e guarda o máximo, o mínimo e a média das temperaturas registadas
- Operações reconhecidas

```
public void sampleTemperature(double temp)
```

Registar a amostra `temp` na estação

```
public double getMaximum()
```

Consultar a máxima temperatura observada até ao momento

```
public double getMinimum()
```

Consultar a mínima temperatura observada até ao momento

```
public double getAverage()
```

Consultar a média das temperaturas observadas até ao momento

```
public void reset(double temp)
```

Apagar a informação registada na estação, para se iniciarem novas medições. É indicada também uma primeira nova medição.

Jogo da Adivinha

Adivinha o Número Secreto

- Defina em Java uma classe `GuessWhat` cujos objectos são jogos de adivinhar números.
- Programe a sua classe no BlueJ.
- Teste um (ou vários) objectos `GuessWhat`, e verifique se se comportam como se espera.

Jogo da Adivinha

- Ouando um jogo é criado, indica-se o valor mínimo e máximo do número secreto (a adivinhar)
- Operações reconhecidas

`public String` `tryit(`**`int`** `guess)`

O jogador usa este método para apostar que o número secreto é `guess`.

O método `tryit` responde (devolve)

“You win!”	se o número proposto é o certo
“Too high!”	se o número proposto é maior que o secreto
“Too low!”	se o número proposto é menor que o secreto
“Game Over”	se o número proposto não faz sentido ... Pois já foi excluído por jogadas anteriores

`public void` `newGame()`

Iniciar novo jogo

Jogo da Adivinha

```
GuessWhat game = new GuessWhat(0,9);
game.tryit(5)
"Too low!"    (String)
game.tryit(8)
"Too high!"   (String)
game.tryit(9)
"Game Over"   (String)
game.newGame();
game.tryit(5)
"Too high!"   (String)
game.tryit(3)
"Too low!"    (String)
game.tryit(4)
"You win!"    (String)
```

Para programar a classe `GuessWhat` vai necessitar da função

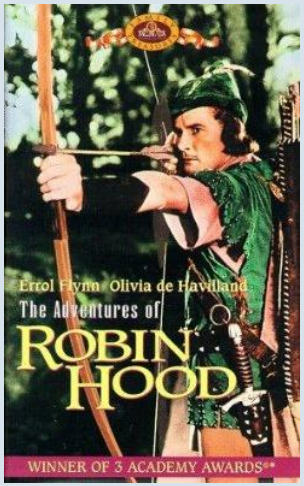
`Math.rand()`

A função `Math.rand()` devolve um número aleatório **`double`** no intervalo [0,1[

Robin Hood (aka, Archer)

Robin Hood (aka Archer)

- Defina em Java uma classe `Archer` cujos objectos são os arqueiros do lendário Robin Hood. Estes arqueiros assaltam os homens do xerife de Nottingham, mas têm de comprar as flechas para os atacar…
- Programe a sua classe no BlueJ.
- Teste um (ou vários) objectos `Archer`, e verifique se se comportam como se espera.



Robin Hood (aka, Archer)

- Cada arqueiro caracteriza-se pelo número de flechas que ainda tem disponível, a sua pontaria (um número real), e pelo dinheiro que lhe resta. O preço das flechas é constante e igual a 5 euros.
- Quando o arqueiro é criado:
  - Se não dissermos nada, o arqueiro começa com 0 flechas, 1.0f de pontaria e 20 euros);
  - Em alternativa, podemos indicar os valores iniciais para o número de flechas, a pontaria e o dinheiro.
- O arqueiro pode disparar flechas contra alvos que se encontram a uma determinada distância (leia-se, contra os malvados cobradores do Xerife de Nottingham):
  - se acertar, recebe um prémio em dinheiro (ou seja, fica com o dinheiro que cobrador do Xerife leva) e aumenta a sua pontaria;
  - se falhar, não aumenta nem o dinheiro nem a pontaria.

Robin Hood (aka, Archer)

- Operações reconhecidas:

```
public int getArrows()
public int getMoney()
public int howManyArrowsCanBuy()
public void buyArrows(int howMany)
public void fireArrow(int distance, int prizeMoney)
public int successfulShot(int distance)
public boolean canFireArrow()
public boolean canContinuePlaying()
```

Robin Hood (aka, Archer)

- Operações reconhecidas (interface de Archer):

`public int` `getArrows()`

Consultar o número de flechas que restam.

`public float` `getAccuracy()`

Consultar a pontaria do arqueiro.

`public int` `getMoney()`

Consultar dinheiro que lhe resta.

`public int` `howManyArrowsCanBuy()`

Consultar quantas flechas tem dinheiro para comprar. Lembre-se que as flechas custam 5 euros cada. Use uma constante, por favor.

Robin Hood (aka, Archer)

- Operações reconhecidas (interface de Archer):

```
public void buyArrows(int howMany)
```

Comprar flechas na quantidade indicada. Cada flecha custa 5 Euros. Se acha estranho que, já no Século XII, as flechas custassem 5 Euros, note que, para além de dinheiro, o Robin Hood também roubava Vinho da Madeira aos homens do Xerife de Nottingham, que o produzia a partir de maçãs dos seus pomares… ☺

```
public int successfulShot(int distance)
```

Calcula o sucesso do tiro. Um tiro tem sucesso se a razão entre a pontaria do arqueiro e a distância for maior ou igual a `0.5f`. A função retorna `0` se o tiro falhou, `1` se acertou. Veja adiante como programar esta operação.

Robin Hood (aka, Archer)

- Operações reconhecidas (interface de Archer):

`public void` `fireArrow(`**`int`** `distance, `**`int`** `prizeMoney)`

Dispara a flecha a uma determinada distância do alvo. Cada tiro custa 1 flecha. Para saber se o tiro é ou não certeiro, use a operação `successfulShot()`. Se acertar:

- Adiciona o prémio, em dinheiro, à sua bolsa
- Aumenta a sua pontaria em metade da distância para o alvo

Quer acerte, quer falhe, gasta sempre uma flecha

`public boolean` `canFireArrow()`

Indica se o arqueiro ainda tem flechas para disparar.

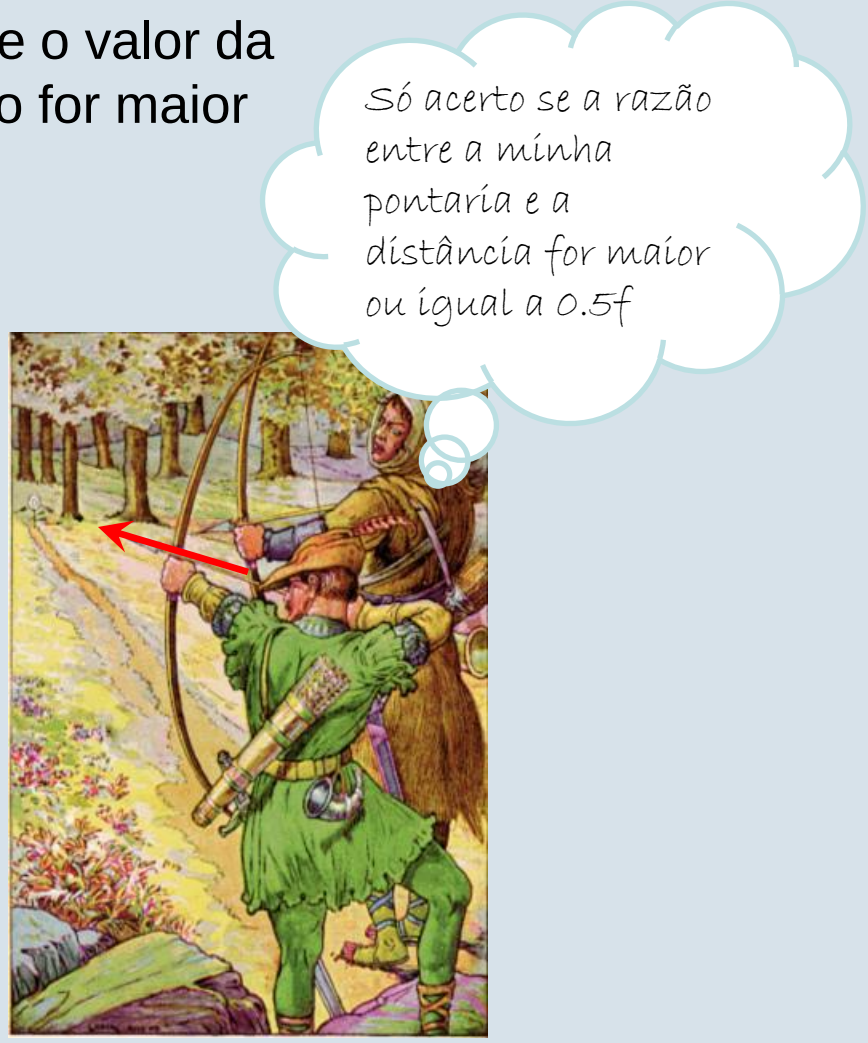
`public boolean` `canContinuePlaying()`

Indica se o arqueiro pode continuar na sua actividade de salteador. A função deve retornar `true` se o arqueiro ainda tiver flechas, ou dinheiro suficiente para as comprar. Caso contrário, retorna `false`.

Robin Hood (aka, Archer)

- `public int successfulShot(int distance)`
  - O arqueiro só acerta se a razão entre o valor da sua pontaria e a distância até ao alvo for maior ou igual a 0.5f
  - A função retorna:
    - `0` se o tiro falhou
    - `1` se acertou.

- Usar:
  - operações aritméticas
  - `Math.round(expr)`
  - `Math.min(expr1, expr2)`
    - devolve o mínimo entre as duas expressões



Robin Hood (aka, Archer)

```
Archer robin = new Archer();
robin.getArrows()
0   (int)
robin.getAccuracy()
1.0   (float)
robin.getMoney()
20   (int)
robin.buyArrows(4);
robin.canContinuePlaying()
true   (boolean)
robin.fireArrow(2000,3);
robin.getArrows()
3   (int)
robin.getAccuracy()
1.0   (float)
robin.fireArrow(1000,3);
robin.getAccuracy()
1.0   (float)
robin.fireArrow(100,3);
robin.getAccuracy()
1.0   (float)
robin.fireArrow(1,3);
robin.getAccuracy()
1.5   (float)
robin.getArrows()
0   (int)
robin.getMoney()
3   (int)
robin.canContinuePlaying()
false   (boolean)
robin.canFireArrow()
false   (boolean)
robin.successfulShot(2)
1   (int)
robin.successfulShot(4)
0   (int)
```