IP – Semana 2 – Lista de Problemas (alunos em 1ª inscrição)

Miguel Goulão

miguel.goulao@di.fct.unl.pt

Objectivos

- No final da 2ª 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

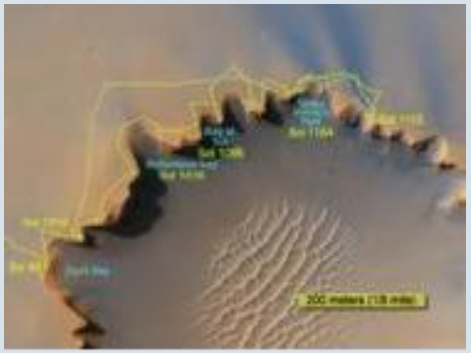
Especificação de interfaces

MARS ROVER
CÍRCULO

Mars Rover

Mars Rover

- Defina em Java uma classe MarsRover cujos objectos têm a funcionalidade indicada.
- Programe a sua classe no BlueJ.
- Teste um (ou vários) objectos MarsRover, e verifique se se comportam como esperado.



Mars Rover

- O MarsRover
  - É um veículo de exploração do planeta Marte.
  - Desloca-se num plano imaginário sobreposto à superfície do planeta, em que cada posição é indicada por um par de coordenadas (números reais).
  - Uma das funcionalidades do MarsRover é medir as distâncias (em linha recta) entre vários pontos no terreno.

Mars Rover

- Cada objecto MarsRover
  - Conhece a sua posição no terreno:
    - A posição é indicada por um par de coordenadas X, Y
  - Conhece a sua direcção de deslocação
    - A direcção é indicada por um ângulo α
- Operações reconhecidas (interface do MarsRover):

```
void moveForward(double distance)
void setHeading(double angle)
void mark()
double getXPos()
double getYPos()
double getHeading()
```

Mars Rover

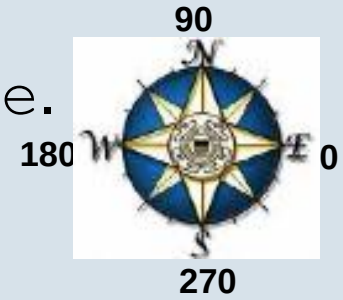
`void` `moveForward`**`(double`** `distance`**`)`**

O Rover avança `distance` unidades na direcção corrente

`void` `setHeading`**`(double`** `angle`**`)`**

O Rover vira-se para a direcção absoluta `angle`.

O ângulo deverá ser indicado em graus.



`double` `getXPos`**`()`**

Consultar o valor da coordenada `X`.

`double` `getYPos`**`()`**

Consultar o valor da coordenada `Y`.

`double` `getHeading`**`()`**

Consultar o valor da orientação do Rover.

Mars Rover

`void` `mark()`

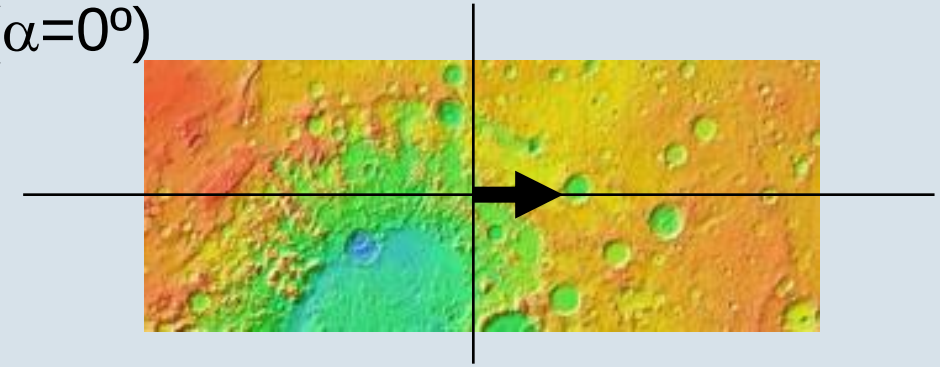
O Rover regista o ponto corrente como ponto base

`double` `getDistance()`

O Rover indica a distância (em linha recta) entre o último ponto base marcado e a sua posição corrente.

Quando é criado, cada objecto MarsRover

- Encontra-se na origem do plano (X=0,Y=0)
- Encontra-se virado para leste (α=0º)



Mars Rover

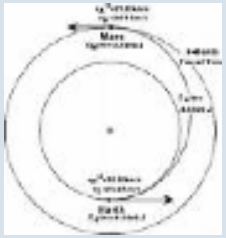
```
MarsRover r = new MarsRover();
r.getXPos()
0.0    (double)
r.getYPos()
0.0    (double)
r.setHeading(90);
r.moveForward(1.0);
r.getXPos()
6.123233995736766E-17  (double)
r.getYPos()
1.0    (double)
r.moveForward(-2);
```

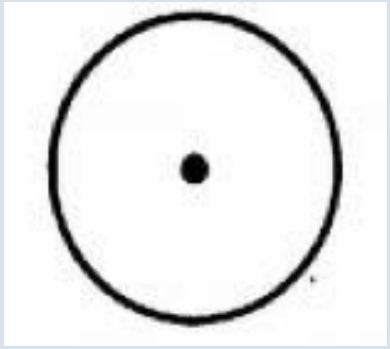
```
r.getYPos()
-1.0    (double)
r.getXPos()
-6.123233995736766E-17  (double)
r.mark();
r.moveForward(2);
r.getDistance()
2.0    (double)
r.setHeading(0);
r.moveForward(2);
r.getDistance()
2.8284271247461903    (double)
```

Círculo

Círculo

- Defina em Java uma classe Circle cujos objectos têm a funcionalidade indicada.
- Programe a sua classe no BlueJ.
- Teste um (ou vários) objectos Circle, e verifique se se comportam como esperado.





Círculo

- Um círculo define-se por
  - Um ponto no plano (par de coordenadas X, Y) indicando o centro
  - O comprimento do raio
- Pretende-se construir círculos de várias maneiras:
  - Dados o seu centro e raio
  - Dado o seu raio
    - centro será na origem do plano
  - Sem indicar nada
    - centro será na origem do plano e o raio será unitário

Círculo

- Operações reconhecidas (interface do Circle):

```
double getPerimeter()
double getArea()
double getRadius()
double getXCenter()
double getYCenter()
boolean ptInCircle(double x, double y)
void translate(double dx, double dy)
```

Círculo

double `getPerimeter()`

Devolve o perímetro do círculo

double `getArea()`

Devolve a área do círculo

double `getRadius()`

Devolve o comprimento do raio do círculo

double `getXCenter()`

Devolve a abcissa do centro

double `getYCenter()`

Devolve a ordenada do centro

boolean `ptInCircle(double x, double y)`

Indica se o ponto (x,y) se encontra no círculo

void `translate(double dx, double dy)`

Desloca o círculo segundo o vector <dx,dy>

Círculo

```
Circle c = new Circle();
c.getRadius()
1.0    (double)
c.getArea()
3.141592653589793    (double)
c.translate(1,1);
c.getArea()
3.141592653589793    (double)
c.ptInCircle(-1,-1)
false    (boolean)
c.getXCenter()
1.0    (double)
c.getYCenter()
1.0    (double)
```

```
Circle d = new Circle(1,1,10);
d.getArea()
314.1592653589793    (double)
d.translate(1,2);
d.getXCenter()
2.0    (double)
d.getYCenter()
3.0    (double)
d.getArea()
314.1592653589793    (double)
Circle e = new Circle(5);
e.getRadius()
5.0    (double)
```

Múltiplos Construtores

```
class Circle {
    ... ;
    Circle() {...; }
    Circle(double radius) { ... ; }
    Circle(double cx, double cy, double radius) {
       ... ;
    }
    ...
}
```

- É possível definir na mesma classe quantos construtores quisermos, para cobrir as várias maneiras pretendidas de criar os objectos.
- Todos os construtores têm o mesmo nome (o nome da classe) mas são distinguidos pelos parâmetros que possuem