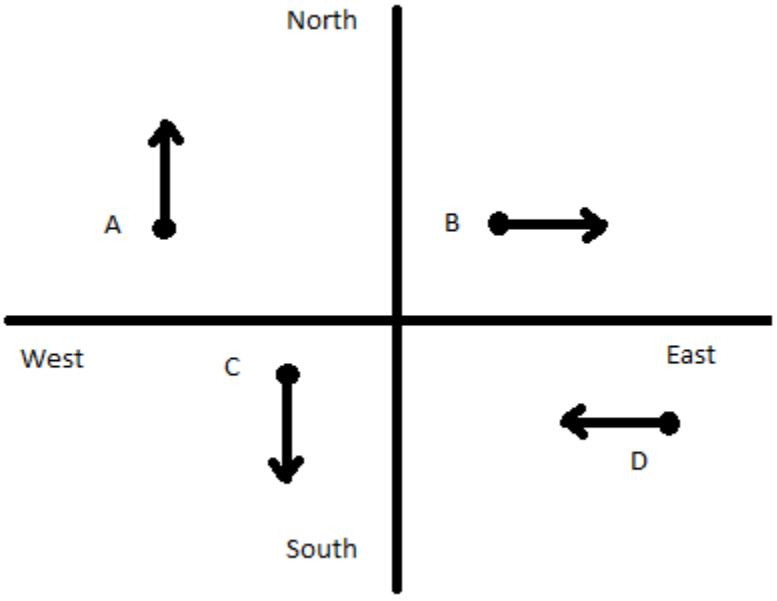
1º Trabalho para casa da unidade curricular de Introdução à Programação, 2010/2011

(27/10/2010 @ 16h28)

Regras: Este trabalho é **individual** e realizado ao abrigo do código de ética dos estudantes do DI, disponível na página de IP. O trabalho é entregue em papel.
Data de entrega: 3 de Novembro de 2010, até às 15h00, na secretaria do DI.

Enunciado

Neste trabalho, pretendemos uma classe Robot, para representar objectos do tipo Robot, que devem ser capazes de guardar a sua *posição*, *orientação*, e *rapidez*. Quando um Robot é criado, podemos ter uma de duas situações: ou não há informação adicional e o Robot é criado na origem dos eixos de coordenadas, com *rapidez* 0.0, e *orientação* a **Norte**, ou é criado recebendo como argumentos as coordenadas da *posição*, a *rapidez* inicial e a *orientação* inicial. Além disso, os Robots devem poder acelerar e travar, bem como mudar a sua *orientação*. Para simplificar, considere que existem apenas quatro *orientações*: Norte, Sul, Este e Oeste. Considere que as coordenadas da *posição* do Robot e a sua *rapidez* são expressas em números reais. A *rapidez* é sempre um número real positivo ou nulo. Por exemplo, os 4 Robots da figura abaixo, identificados pelas letras A, B, C e D, têm todos a mesma *rapidez* (por exemplo, 2 m/s), mas orientações distintas (Norte, Este, Sul e Oeste, respectivamente). Naturalmente, Robots diferentes podem ter valores de *rapidez* e *orientação* diferentes, e quer a *rapidez* quer a *orientação* podem mudar ao longo da vida do Robot.



A sua classe deverá ainda conter uma operação para movimentar o Robot durante um determinado período de tempo e outra para, dadas a sua posição, orientação e rapidez actuais, determinar se um determinado ponto está ou não na sua rota (independentemente do tempo que lá leva a chegar). Passamos a descrever um pouco mais detalhadamente cada uma das operações:

```java
/*
 * o robot vira-se para Norte, mantendo a rapidez
 */
public void north()
/*
 * o robot vira-se para Sul, mantendo a rapidez
 */
public void south()

/*
 * o robot vira-se para este, mantendo a rapidez
 */
public void east()

/*
 * o robot vira-se para oeste, mantendo a rapidez
 */
public void west()

/*
 * soma acel à rapidez actual mantendo a orientação
 * @pre acel > 0, ou seja, não há acelerações negativas
 * @param acel aceleração a aplicar à rapidez
 */
public void increaseSpeed(double acel)

/*
 * subtrai brk à rapidez actual mantendo a orientação
 * @pre brk <= rapidez, ou seja, rapidez nunca fica negativa
 * @param brk travagem a aplicar à rapidez
 */
public void decreaseSpeed(double brk)

/*
 * devolve a coordenada X da posição
 */
public double posX()

/*
 * @return a coordenada Y da posição
 */
public double posY()

/*
 * @return a rapidez actual
 */
public double speed()

/*
 * @return a orientação actual: (1 - Norte, 2 - Sul, 3 - Este, 4 - Oeste)
 */
public int orientation()

/*
 * desloca o robot na orientação actual, à rapidez actual, durante seconds.
 * Por exemplo, se o robot estiver na posição (2.4, 5.0), virado para norte,
 * com rapidez 1.5, e seconds valer 2, depois desta operação deve ficar na posição
 * (2.4, 8.0)
 * @param seconds o número de unidades de tempo que o movimento vai durar
 */
public void move(int seconds)

/*
 * determina se o robot está em rota de colisão com o ponto (x, y) que atingirá ao fim
 * de algum tempo (não importa quanto tempo). Por exemplo, um Robot na posição
 * (2.0, 4.0), a deslocar-se para Este, com uma rapidez 3.0 acabará por passar pelo
 * ponto (9.0, 4.0), mas não pelos pontos (2.0, 8.0), (-3.0, 4.0), ou (16.9, -4.7).
 * @param x coordenada x do ponto a testar
 * @param y coordenada y do ponto a testar
 * @return true, se o ponto fica a caminho, ou false, caso contrário
 */
public boolean onRouteTo(double x, double y)
```

Programe em Java a classe Robot.

Material a entregar

Documento, em papel, com as seguintes secções:

1. Identificação do aluno, com o nome completo, número de aluno, turno prático em que está inscrito (P1, P2, ..., P8) e o nome do docente do turno prático.

2. Listagem, manuscrita ou impressa, da sua classe Robot. Deverá definir o conjunto de operações (métodos e construtores), variáveis e constantes que considerar conveniente para esta classe.

Exemplo de traço de execução no BlueJ

```
Robot r = new Robot();
r.posX()
0.0   (double)
r.posicaoY()
0.0   (double)
r.increaseSpeed(1.0);
r.posX()
0.0   (double)
r.posY()
0.0   (double)
r.speed()
1.0   (double)
r.move(1);
r.posY()
1.0   (double)
r.increaseSpeed(2.0);
r.move(1);
r.posX()
0.0   (double)
r.posY()
4.0   (double)
r.speed()
3.0   (double)
r.decreaseSpeed(2.0);
r.speed()
1.0   (double)
r.east();
r.speed()
1.0   (double)
r.orientation()
3   (int)
r.move(1);
r.posX()
1.0   (double)
r.posY()
4.0   (double)
r.south();
r.move(1);
r.posX()
1.0   (double)
r.posY()
3.0   (double)
r.west();
r.increaseSpeed(1.3);
r.speed()
2.3   (double)
r.move(2);
r.posX()
-3.6   (double)
r.posY()
3.0   (double)
r.onRouteTo(-20.3, 3.0)
true   (boolean)
```

Erros frequentes a evitar:

- Variáveis desnecessárias
- Declaração de variáveis de instância públicas, ou de métodos da interface do objecto privados
- Código muito confuso
- Identificadores (de classes, constantes, variáveis, métodos, parâmetros) incompreensíveis, ou pouco expressivos
- Inexistência de constantes, ou não utilização das existentes
- Ausência de comentários (que fazem mesmo falta)