Universidade Estadual de Campinas
Instituto de Computação

Laboratório 2

Simulador de Robôs

MC322 - Programação Orientada a Objetos

Professora: Esther Colombini
PEDs: Angelica Cunha dos Santo / Athyrson Machado Ribeiro /
Wladimir Arturo Garces Carrillo

---

1. Descrição Geral

Um simulador de robôs é um ambiente computacional que permite modelar, testar e analisar o comportamento de robôs em diferentes cenários sem a necessidade de hardware físico. Esses simuladores são amplamente utilizados na pesquisa, na indústria e na educação para desenvolver e validar algoritmos de controle, navegação, inteligência artificial e interação com o ambiente antes da implementação real.

A simulação de robôs possibilita testar e aperfeiçoar sistemas em condições seguras, reduzindo custos e riscos associados ao desenvolvimento de robôs físicos. Além disso, permite explorar diferentes estratégias de movimentação, sensores, e tomada de decisão em cenários variados. Na era do aprendizado de máquina, simuladores são imprescindíveis para o treinamento de robôs antes da implementação em agentes reais.

1.1. Principais Componentes de um Simulador de Robôs

- **Modelo do Robô:** Representação digital do robô, incluindo sua estrutura física, mecanismos de locomoção e atuadores. Pode ser um robô simples com rodas ou um modelo mais complexo, como um robô humanoide.
- **Ambiente Virtual:** Define o espaço onde o robô operará (plano cartesiano, terreno 3D, fábrica, casa, etc.). Pode incluir obstáculos, pontos de referência e interações físicas (colisões, gravidade).
- **Mecanismos de Movimentação:** Simulam os sistemas de locomoção do robô, como rodas, pernas ou esteiras. Podem incluir restrições realistas, como atrito e aceleração.
- **Sensores e Percepção:** Sensores simulados (câmeras, sensores de distância, giroscópios, GPS, LiDAR). Permitem que o robô "perceba" o ambiente ao seu redor.

2. Objetivo

Este laboratório tem como objetivo implementar um sistema de simulação de robôs que se movimentam em um ambiente bidimensional. Os robôs são classificados em **terrestres** e **aéreos**, cada um com características e comportamentos específicos. O laboratório visa aplicar conceitos de **herança**, **sobrescrita de métodos** e **polimorfismo** para modelar os diferentes tipos de robôs e **sobrecarga de métodos** e **relações entre classes** para suas interações no ambiente.

3. Descrição das Classes

Agora os robôs serão classificados em **terrestres** e **aéreos**, com características especificas.

3.1. Classe Base `Robo`

A classe `Robô` é a **Classe Base** para todos os tipos de robôs. Ela contém os seguintes atributos e métodos:

- **Atributos:**
  - `nome` (String) → Nome do robô.
  - `direcao` (String) → Direção atual do robô (Norte, Leste, Sul, Oeste).
  - `posicaoX` (int) → Coordenada em X do robô no ambiente.

- `posicaoY` (int) → Coordenada Y do robô no ambiente.

- **Métodos:**
  - `mover(int deltaX, int deltaY)` → Move o robô para uma nova posição.
  - `identificarObstaculo()` → Identifica os obstáculos ao redor do robô no ambiente.
  - `exibirPosicao()` → Exibe a posição atual do robô.

3.2. Robôs Terrestres

Os robôs terrestres herdam da classe `Robo` e possuem características específicas:

3.2.1. Classe `RoboTerrestre` (Herdada de `Robo`):

- **Atributo adicional:**
  - `velocidadeMaxima` (int) → Define a velocidade máxima do robô.
- **Método sobrescrito:**
  - `mover()` → Move o robô se a velocidade do movimento não excede a `velocidadeMaxima`.
- **Nota:**
  Crie, pelo menos, duas classes adicionais que herdem da classe `RoboTerrestre` (Tipos diferentes de Robôs Terrestre). Cada classe deve ter pelo menos um atributo e um método especifico que justifique sua criação, por exemplo, movimentação diferenciada.

3.3. Robôs Aéreos

Os robôs aéreos herdam da classe `Robo` e possuem características específicas:

3.3.1. Classe `RoboAereo` (Herdad de `Robo`):

- **Atributos adicionais:**
  - `altitude` (int) → Altura atual que o robô pode alcançar.
  - `altitudeMaxima` (int) → Altura máxima que o robô pode alcançar.
- **Métodos adicionais:**
  - `subir(int metros)` → Aumenta a altitude do robô.
  - `descer(int metros)` → Reduz a altitude do robô.
- **Nota:**
  Crie, pelo menos, duas classes adicionais que herdem da classe `RoboAereo` (Tipos diferentes de Robôs Aéreos). Cada classe deve ter pelo menos um atributo e um método especifico que justifique sua criação, por exemplo, movimentação diferenciada.

3.4. Classe `Ambiente`

A classe `Ambiente` representa o espaço onde os robôs se movimentam. Pegue a classe `Ambiente` do lab 1 e faça as melhoras:

- Adicionar uma lista de robôs ativos (`ArrayList<Robo>`.
- Criar método `adicionarRobo(Robo r)`.
- Modificar `dentroDosLimites() para considerar altura (Para robôs aéreos).`

4. Atividades

1. **Implementação das Classes:**
   - Implemente as classes `Robo`, `RoboTerrestre`, `RoboAereo` e suas respectivas classes herdeiras e `Ambiente` conforme descrito acima.
2. **Testes na Classe `Main`:**
   - Crie um ambiente.
   - Adicione os diferentes tipos de robôs ao ambiente.
   - Teste os seguintes cenários:
     - Movimentação terrestre e aérea.
     - Limites de velocidade (`RoboTerrestre`).
     - Altura máxima (`RoboAereo`).
   - Teste os métodos específicos das classes herdeiras que vocês criaram.
   - Exiba a posição final de cada robô no console.
3. **Melhorias e Extensões:**
   - Adicione validações para garantir que os robôs não se movam para coordenadas negativas.

5. Considerações Finais

Este laboratório permite explorar conceitos avançados de programação orientada a objetos, como herança, polimorfismo e sobrescrita de métodos, além de reforçar a importância de um design de classes bem estruturado. A implementação em Java é apenas um exemplo, mas o mesmo conceito pode ser aplicado em outras linguagens orientadas a objetos.

6. Entrega e Avaliação

Para a entrega do trabalho considere:

1. **A entrega do Laboratório é realizada exclusivamente via Github**[1].
2. Para a submissão, no Github gere um release (tag) com a identificação do laboratório no estilo *"lab01-RA1-RA2"*. Por exemplo, para a dupla com alunos com RA 123456 e 987654 a tag será: lab02-123456-987654.
3. Observação: Evite criar releases enquanto não tiver certeza que seu código está funcionando como esperado.
4. Utilize os horários de laboratório e atendimentos para tirar eventuais dúvidas de submissão e também relacionadas ao desenvolvimento do laboratório.
5. **Prazo de Entrega:** 30/03/25 - 23h59.

O código será avaliado com base nos seguintes critérios:

- Uso correto de POO (herança, polimorfismo, sobrescrita de métodos).
- Boas práticas (encapsulamento, modularidade, gets e sets, comentários no código).
- Implementação correta das funcionalidades.
- Interação funcional entre robôs e ambiente.
- Criatividade na implementação das classes e suas funcionalidades.

[1] Você deve criar um link da release e enviar no Google Classroom