Classes Abstratas

Classes Abstratas

- Classes abstratas: não podem ser instanciadas
  - Poderoso Mecanismo de Abstração:

    Permite a herança do código sem violar a noção de subtipo

    Diz *o que* deve ter a subclasse, mas não diz *como* !
  - A classe abstrata:
    - código genérico, livre de particularidades
  - As subclasses:
    - detalhes particulares

Classes Abstratas

Métodos Abstratos:

- Só a assinatura, sem corpo
- Precisam ser implementados pelas subclasses (folhas)

- A classe abstrata enumera características genéricas do modelo, mas não as implementa.
- A classe abstrata obriga subclasses (folhas) a implementarem funcionalidades abstratas previstas em seu corpo.
- Para isto, cada subclasse se utiliza de seus detalhes particulares.

Classes Abstratas

Exemplo:

Círculos, Quadrados e Triângulos.

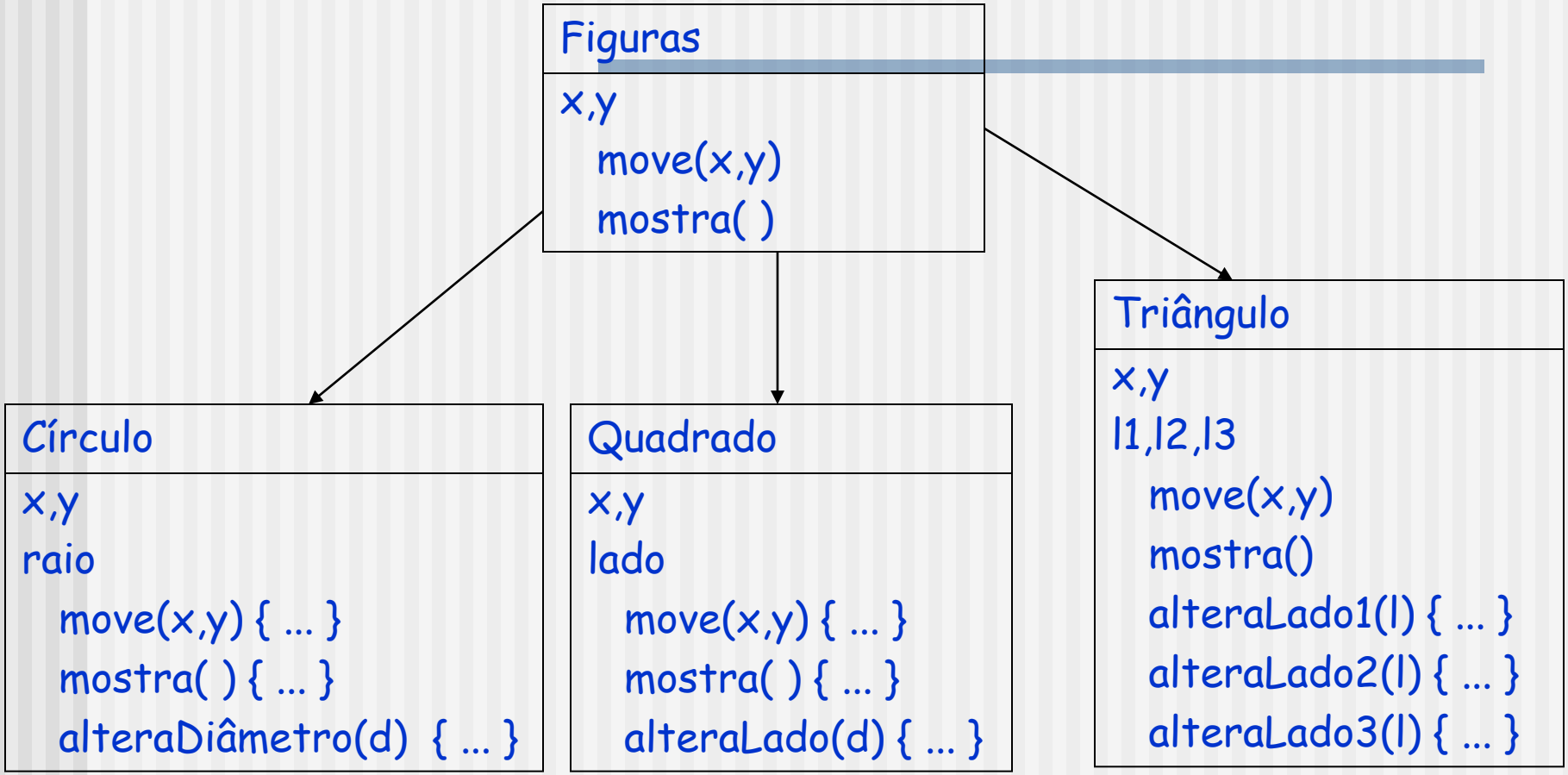
Círculo
x,y raio move(x,y) { ... } mostra( ) { ... } alteraDiâmetro(d) { ... }

Quadrado
x,y lado move(x,y) { ... } mostra( ) { ... } alteraLado(d) { ... }

Triângulo
x,y l1,l2,l3 move(x,y) mostra() alteraLado1(l) { ... } alteraLado2(l) { ... } alteraLado3(l) { ... }

Classes Abstratas

Exemplo: Círculos, Quadrados e Triângulos.



Classes Abstratas

Exemplo:

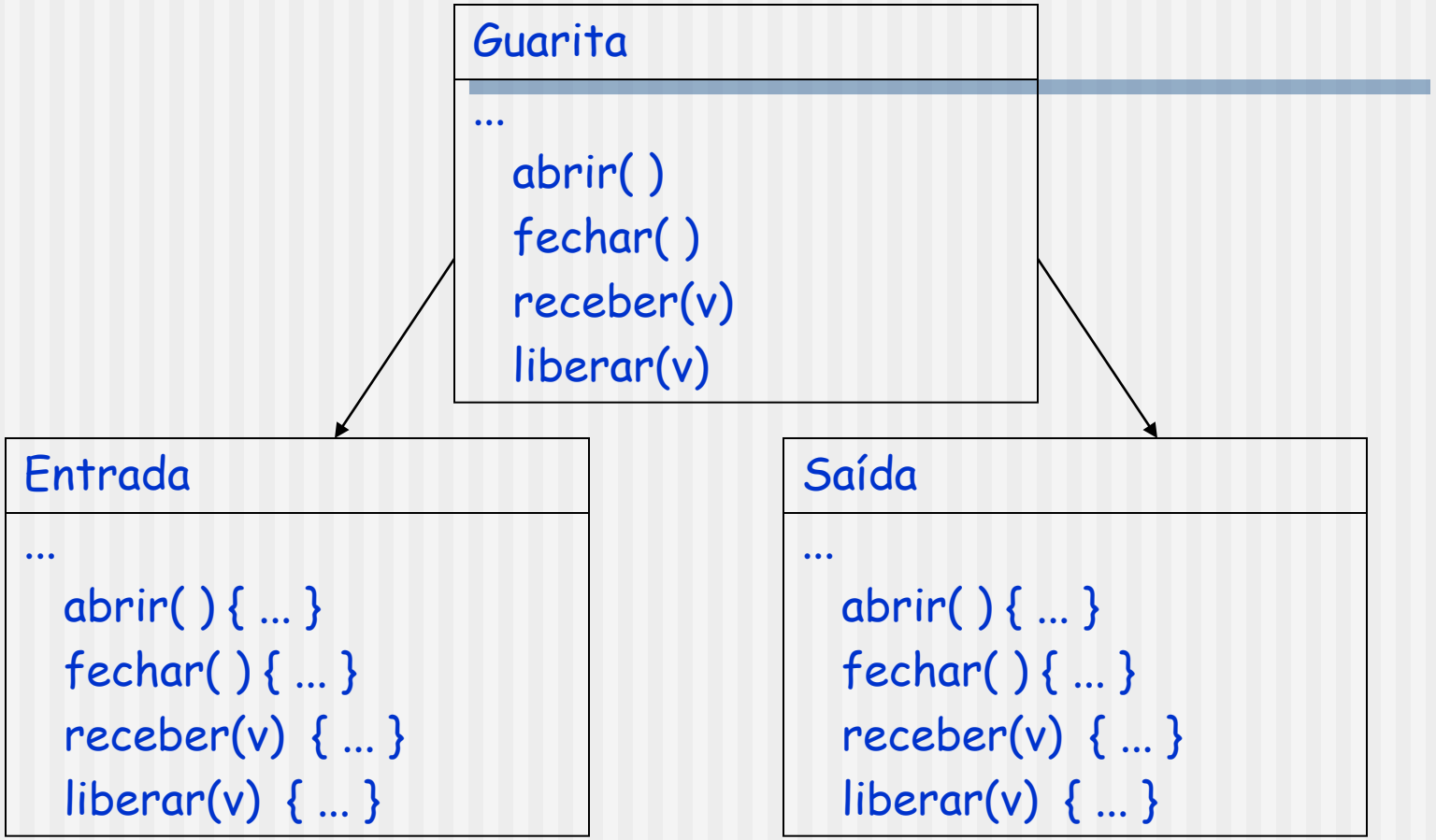
Guarita de entrada, Guarita de saída.

Guarita de entrada
... abrir( ) { ... } fechar( ) { ... } receber(v) { ... } liberar(v) { ... }

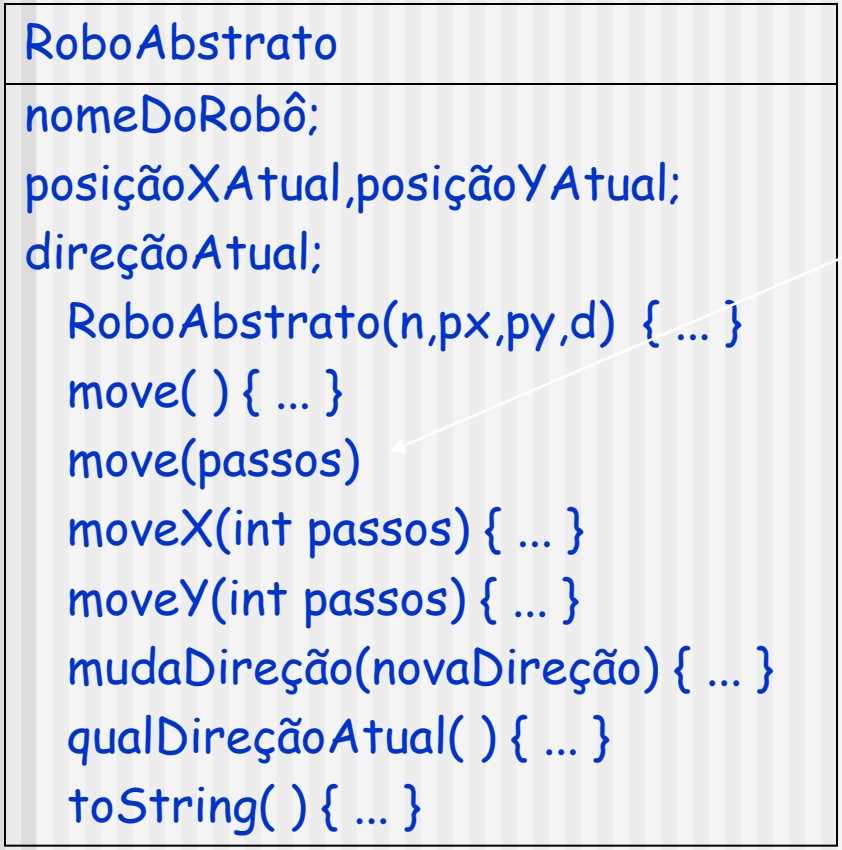
Guarita de saída
... abrir( ) { ... } fechar( ) { ... } receber(v) { ... } liberar(v) { ... }

Classes Abstratas

Exemplo: Guarita de entrada, Guarita de saída.



Classes Abstratas



método abstrato: não sabemos exatamente como um robô irá implementar seus movimentos

Classes Abstratas

```
abstract class RoboAbstrato {
  private String nomeDoRobo;
  private int posiçãoXAtual,posiçãoYAtual;
  private short direçãoAtual;

RoboAbstrato(String n,int px,int py,short d)  { ... }
    public void move()     {  move(1);   }
    public abstract void move(int passos);
    public void moveX(int passos)   { ... }
    public void moveY(int passos)   { ... }
    public void mudaDireção(short novaDireção)    { ... }
    public short qualDireçãoAtual()     { ... }
    public String toString()    { ... }
  }
```

A classe é abstrata, pois possui ao menos um método abstrato.

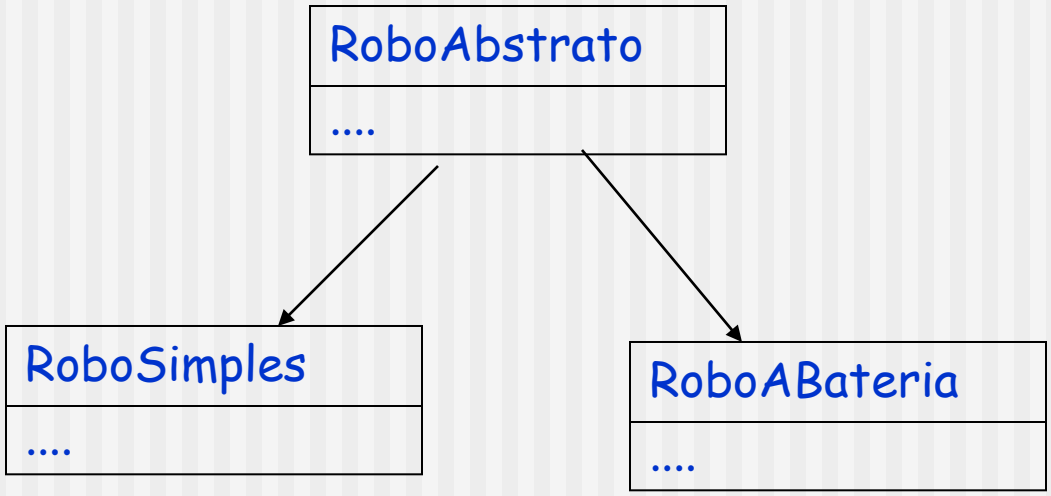
Não existem "campos abstratos".

Um construtor não pode ser abstrato: seu código é necessário para inicializar corretamente os campos da classe abstrata.

Um método abstrato pode ser chamado a partir de outro: no momento da chamada, a subclasse haverá sobreposto o método abstrato.

RoboAbstrato.java

Classes Abstratas



Cada subclasse de RoboAbstrato implementa move(passos) de uma forma particular:

Robô simples apenas atualiza as coordenadas,

Robô a bateria consome energia.

Classes Abstratas

```
class RoboSimples extends RoboAbstrato {
RoboSimples(String n,int px,int py,short d)
   {  super(n,px,py,d);  }

public void move(int passos)
   { switch(qualDireçãoAtual()) {
        case   0:   moveX(+passos); break;
        case  90:  moveY(+passos); break;
        case 180: moveX(-passos); break;
        case 270: moveY(-passos); break;   }
   }
 } // fim da classe RoboSimples
```

RobôSimples herda todas as caracerísticas de RobôAbstrato, mas deve implementar move(passos) para que possa ser instanciada.

RoboSimples.java

Classes Abstratas

```
class RoboABateria extends RoboAbstrato {
private long energia;
RoboABateria(String n,int px,int py,short d,long e)    {
      super(n,px,py,d); energia = e;    }
public void move(int passos)     {
   long energiaASerGasta = passos*10;
   if (energiaASerGasta <= energia) {
        switch(qualDireçãoAtual()) {
          case   0: moveX(+passos); break;
          ......
          case 315: moveY(-passos); moveX(+passos); break;  }
       energia -= energiaASerGasta;
     }
   }
public String toString() { ... }
}
```

Robô que consome energia em seus movimentos

RoboABateria.java

Classes Abstratas

```
class DemoRobos  {
  public static void main(String[] argumentos)   {
    ....
    RoboAbstrato imag = new RoboAbstrato("Imaginário",0,0,(short)180);
    } // fim do método main
  } // fim da classe DemoRobos
```

DemoRobos.java

```
F:\>javac DemoRobos.java
DemoRobos.java:34: RoboAbstrato is abstract; cannot be instantiated
    RoboAbstrato imag = new RoboAbstrato("Imaginßrio",0,0,(short)180);
                        ^
1 error
```

Classes Abstratas

- Exercício:

- Implementar a classe abstrata FiguraPlana, contendo os métodos abstratos area() e perimetro() , e as subclasses Quadrado, Triangulo, Losango e Circulo.

- Testar as novas classes usando um ArrayList<FiguraPlana>. Criar instancias variadas de figuras, adicionar ao arraylist, listar as figuras incluídas no arraylist e totalizar a área e o perímetro delas.