Programação Orientada a Objetos

INTRODUÇÃO

Plataforma Java

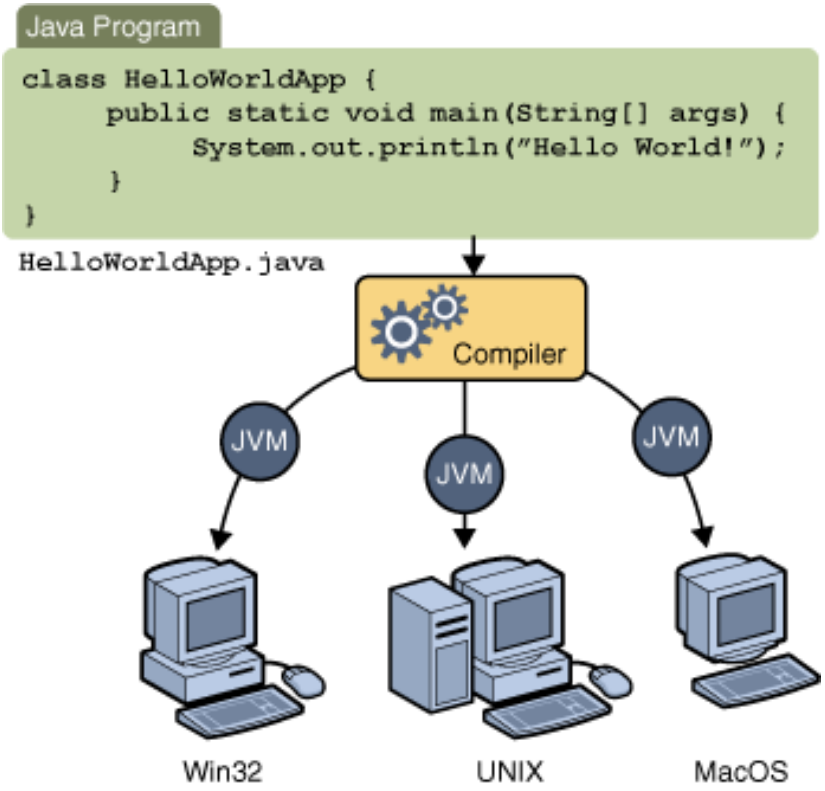
- Java é tanto uma linguagem de programação de alto nível quanto uma plataforma de desenvolvimento de sistemas
- Como linguagem, Java é orientada a objetos, independente de arquitetura (multiplataforma), portável, robusta, segura, interpretada, distribuída, etc

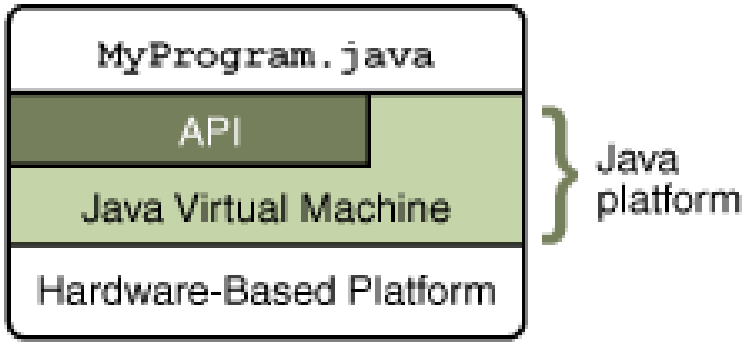
Plataforma Java

- Java SE (Java Platform Standard Edition)
  - Desenvolvimento e execução de applets, aplicações stand-alone ou aplicações cliente
- Java EE (Java Platform Enterprise Edition)
  - Reúne um conjunto de tecnologias em uma arquitetura voltada para o desenvolvimento de aplicações servidoras
- Java ME (Java Platform Micro Edition)
  - Fornece um ambiente de execução otimizado e permite escrever programas cliente que são executados em pequenos dispositivos móveis (smart cards, telefones celulares, ...)

Plataforma Java

- Compilador e máquina virtual disponíveis para vários sistemas operacionais





Introdução à Programação Orientada a Objetos

- O que é um paradigma de programação?
  - É um padrão conceitual que orienta soluções de projeto e implementação
  - Paradigmas explicam como os elementos que compõem um programa são organizados e como interagem entre si
    - Exs.: procedural, funcional, orientado a objetos

Orientação a Objetos

- É baseada na modelagem de objetos do mundo real
- O que é um objeto?
  - Uma entidade que você pode reconhecer
  - Uma abstração de um objeto do mundo real
  - Uma estrutura composta de dados e operações sobre esses dados

Objetos

- Cada objeto possui características (atributos) e comportamento (operações)
  - Ex.: lâmpada
    - características: ligada (sim/não), potência, voltagem
    - comportamento: ligar, desligar, queimar

Objetos

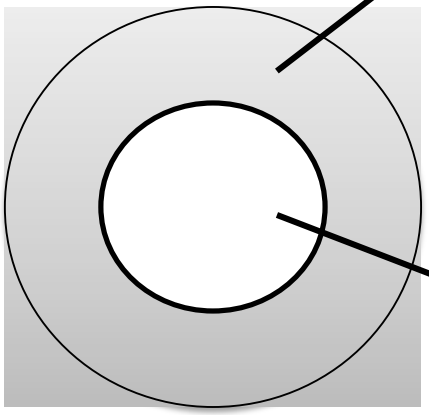
- Um programa orientado a objetos é estruturado como uma comunidade de objetos que interagem entre si
  - Cada objeto tem um papel a cumprir
  - Cada objeto oferece um serviço ou realiza uma ação que é usada por outros objetos
  - Ex.: um objeto Lustre interage com diversos objetos Lâmpada

Classes

- A classe é a definição formal dos atributos e métodos que compõem os objetos
- Objetos são instâncias de uma classe

Encapsulamento

- Encapsular é esconder como as coisas funcionam por trás de uma interface externa
  - Interface são as operações que o objeto fornece para os demais objetos
  - É um dos conceitos básicos da Orientação a Objetos
- A ideia é de uma "caixa preta":
  - Não é necessário saber os detalhes de funcionamento interno do objeto, mas sim como utilizá-lo
- Ex.: caixa automático
  - Como ele é implementado internamente?
  - Utilizamos através de operações bem conhecidas



objeto

Métodos

•Interface pública declara as operações permitidas

Atributos

•Acesso aos atributos é feito através da chamada de um método

Encapsulamento

- Alguns benefícios:
  - A implementação interna de um objeto pode mudar e o resto do sistema não é afetado (desde que a interface de acesso não mude)
  - Maior segurança ao proteger os atributos de um objeto de alterações indevidas por outros objetos
  - Maior independência entre os objetos, pois eles só precisam conhecer a interface externa definida

Projetando Objetos

- De uma forma simples, o projeto orientado a objetos de um sistema pode ser dividido em três etapas:
  - Identificar as abstrações/entidades envolvidas no problema
  - Identificar o comportamento que cada uma destas entidades deve ser capaz de fornecer
  - Identificar os relacionamentos entre essas entidades
  - Identificar as estruturas de dados internas necessárias para implementar o comportamento e relacionamentos desejado

Diagramas UML

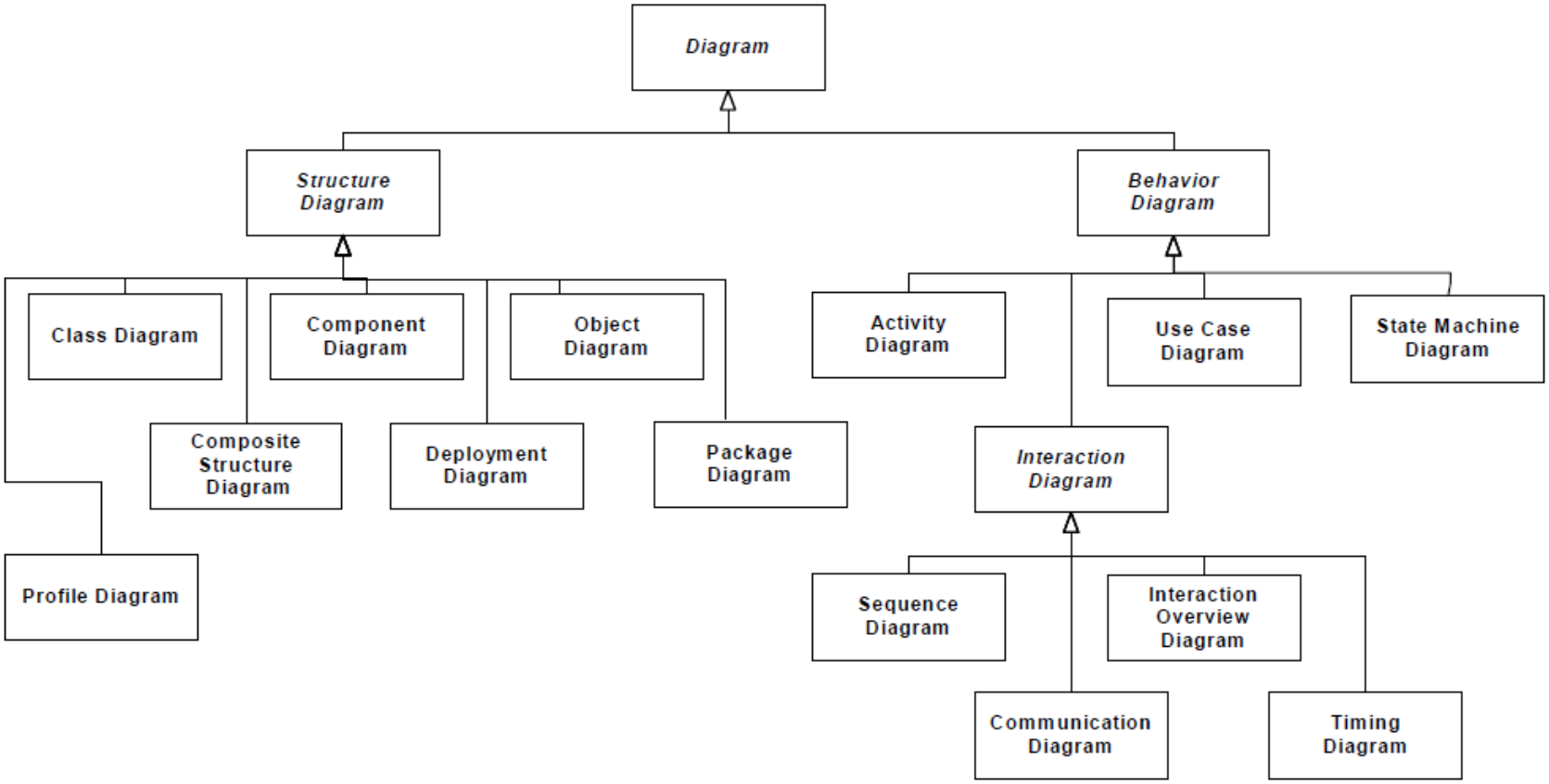


Diagrama de Classes UML

- Denota a estrutura estática do sistema
- Apresenta as classes e seu relacionamentos com outras classes

Diagrama de Classes da UML

Diagrama de Classes da UML

- Modificadores:
  - Público +
  - Privado -

Diagrama de Classes UML

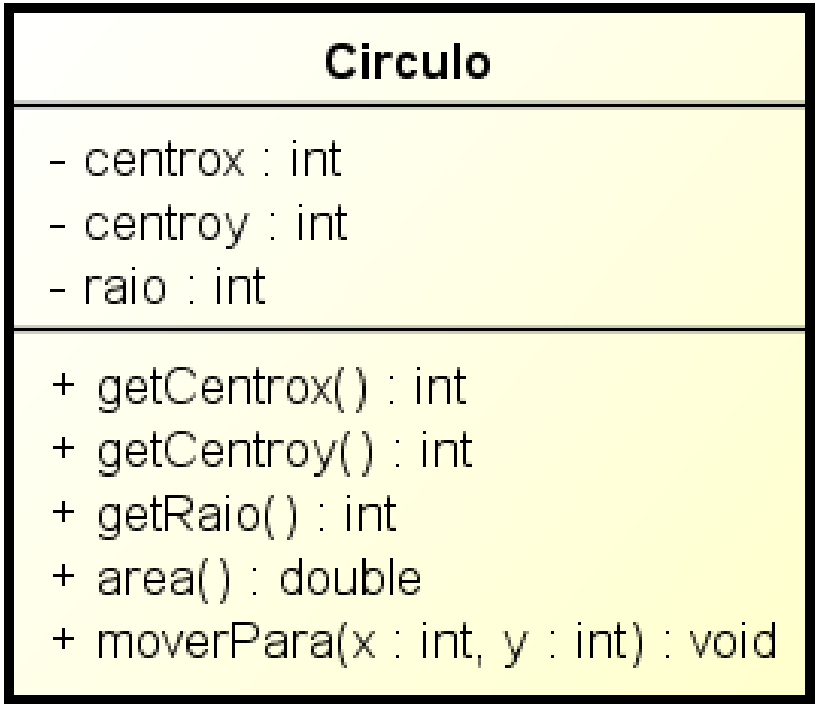


Diagrama de Classes UML

- Relacionamento de dependência:
  - É um relacionamento que significa que um elemento necessita de outro elemento para sua especificação ou implementação
  - É um relacionamento “fornecedor-cliente”
    - Um objeto fornece algo que outro objeto utiliza

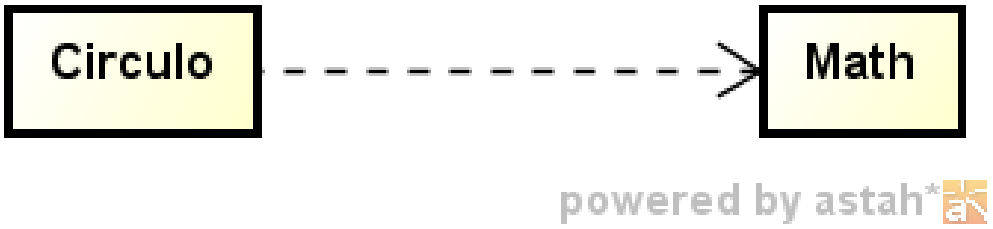


Diagrama de Classes UML

- Relacionamento de associação:
  - É um relacionamento estrutural que descreve um conjunto de ligações, onde uma ligação é uma conexão entre objetos
  - Usualmente implementado através de atributos

Diagrama de Classes UML

- Relacionamento de associação:
  - Navegabilidade da associação
    - Bidirecional

    - Unidirecional

Diagrama de Classes UML

- Relacionamento de associação:
  - Multiplicidade da associação
    - Especifica-se o menor e o maior valor
    - Formato Menor..Maior
    - Valores mais utilizados
      - Menor: 0 (opcional), 1 (obrigatório)
      - Maior: 1 (somente um), * (vários)

Diagrama de Classes UML

- Relacionamento de associação:
  - Multiplicidade da associação
    - Cliente tem uma única conta (1..1 ou 1)

    - Cliente pode ter ou não uma conta

    - Cliente tem várias contas, mas no mínimo uma

    - Cliente tem várias contas, mas não é obrigatório (0..* ou *)

Resumo

- Objeto
  - Unidade básica de orientação a objetos. Um objeto é uma entidade que tem atributos, comportamento e identidade. Objetos são membros de uma classe e os atributos e métodos de um objeto são definidos pela classe.
- Classe
  - Uma classe é uma descrição de um conjunto de objetos. Este conjunto de objetos compartilha atributos e comportamento em comum. Uma definição de classe descreve todos os atributos dos objetos membros da classe, bem como os métodos que implementam o comportamento destes membros.

Resumo

- Orientação a objetos
  - Um paradigma de programação que usa abstração com objetos, classes encapsuladas e comunicação por mensagens, hierarquia de classes e polimorfismo.
- Abstração
  - Um modelo de um conceito ou objeto do mundo real.
- Encapsulamento
  - Processo de esconder os detalhes internos de um objeto do mundo externo.

Resumo

- Comportamento
  - Atividade de um objeto que é vista do ponto de vista do mundo externo. Inclui como um objeto responde a mensagens alterando seu estado interno ou retornando informação sobre seu estado interno.
- Método
  - Uma operação ou serviço executado sobre o objeto, declarado como parte da estrutura da classe. Métodos são usados para implementar o comportamento do objeto.
- Estado
  - Reflete os valores correntes de todos os atributos de um objeto e são o resultado do comportamento do objeto ao longo do tempo.
- Atributo
  - Usado para armazenar o estado de um objeto. Pode ser simples como uma variável escalar (int, char, double, ou boolean) ou pode ser uma estrutura complexa tal como outro objeto.

PROGRAMAÇÃO COM JAVA

Estrutura de um Programa

- Um programa Java é um conjunto composto por uma ou mais classes
- Tipicamente, cada classe é implementada em um arquivo fonte separado, sendo que o arquivo deve ter o mesmo nome da classe.
  - Ex.: a classe Lampada deve estar definida no arquivo Lampada.java
- Em geral, os arquivos que compõem um programa java devem estar no mesmo diretório

Biblioteca de Classes (API)

- Application Programming Interface
- É uma coleção de classes, normalmente provendo uma série de facilidades que podem ser usadas em programas
- Classes são agrupadas em conjuntos chamados packages
  - Exs:
  - java.lang: inclui classes básicas, manipulação de arrays e strings. Este pacote é carregado automaticamente pelo programa
  - java.io: operações de input e output
  - java.util: classes diversas para manipulação de dados

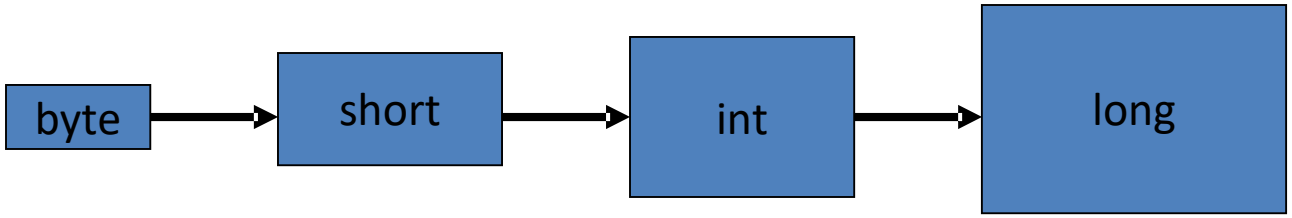
Tipos de Dados Básicos

- Tipos de dados primitivos
  - inteiros: byte (8 bits), short (16), int (32), long (64)
    - 1 (decimal) , 07 (octal), 0xff (hexadecimal), 1L(long)
  - reais: float (32), double (64)
    - 3.0F (float), 4.02E23 (double), 3.0 (double)
  - caractere: char (16)
    - ‘a’, ‘\141’, ‘\u0061’, ‘\n’
  - booleano: boolean (8)
    - true, false

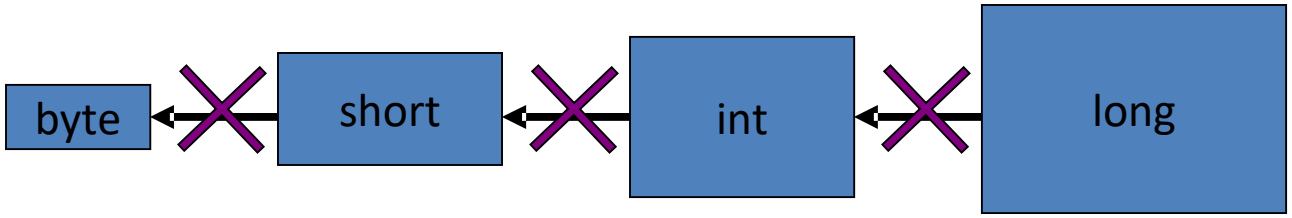
Tipos de Dados Básicos

- Em Java, tem-se dois tipos de conversão de valores:
  - conversão para um tipo maior
    - automática
  - conversão para um tipo menor (chamada de *casting*)
    - não é automática

```
int x=1;
long y=x;
```




```
long y=1;
int x=y;
```
Erro!!!



Tipos de Dados Básicos

- Para converter de um tipo para um tipo menor, precisamos referenciar de forma explícita.
  - *(tipo Java) expressão;*
  - Ex.:
    - ```
      long y = 1;
      int x = (int)y;
      ```
    - ```
      byte b1=1, b2=2, b3;
      b3 = (byte)(b1 + b2);
      ```
  - Cuidado! Ao somar dois valores *byte* iguais a 100, o resultado é o *int* 200. Ao realizar o *cast* para *byte*, o resultado é convertido para -56, o equivalente ao padrão de bits armazenados.

Operadores

- Operadores básicos:
  - aritméticos: +, -, *, /, % (resto da divisão)
  - relacionais: >, >=, <, <=
  - igualdade: ==, !=
  - lógicos: &&, & (and), ||, | (or), ^ (xor), ! (not)
  - atribuição: =, +=, -=, *=, /=, %=
  - incremento, decremento: ++, --

Operadores

- A maioria dos operador aritméticos resultam em *int* ou *long*
  - Quando utilizamos valores *byte* e *short*, eles são convertidos para *int* antes da operação
  - Da mesma forma, se um dos operandos for *long*, os outros são convertidos para *long* antes da operação
  - Ex.:
    - 10 + 10 o resultado é *int*
    - 10L + 10 o resultado é *long*

Operadores

- Cuidado:
  - O resultado da operação de divisão em Java depende do tipo dos operandos
    - Tipo inteiro: o resultado é a divisão inteira

```
int resultado = 10/4 //igual a 2
```

    - Tipo ponto flutuante: o resultado é a divisão decimal

```
float resultado = 10f/4f //igual a 2.5
```

Funções Matemáticas

- Funções matemáticas (classe *Math*):
  - sqrt(x): cálculo da raiz quadrada de x (x é do tipo double)
  - abs(x): valor absoluto de x (x pode ser float, int, long)
  - cos(x): coseno trigonométrico de x (x em radianos)
  - exp(x): método exponencial e^x
  - pow(x,y): x elevado a potência y (x^y)
- Exemplo:

```
double raio;
raio = Math.sqrt(area/Math.PI);
```