Programação Orientada a Objetos

UNIVERSIDADE DO ESTADO DE MINAS GERAIS | UEMG

GitHub

https://github.com/mauro-hemerly/UEMG-2019-1/POO

Java 11

Mauro Hemerly (Hämmerli) Gazzani
mauro.hemerly@gmail.com

NetBeans

Programação Orientada a Objetos

IDEs e Java online

NetBeans eclipse IntelliJIDEA

IDE (integrated Development Environment)

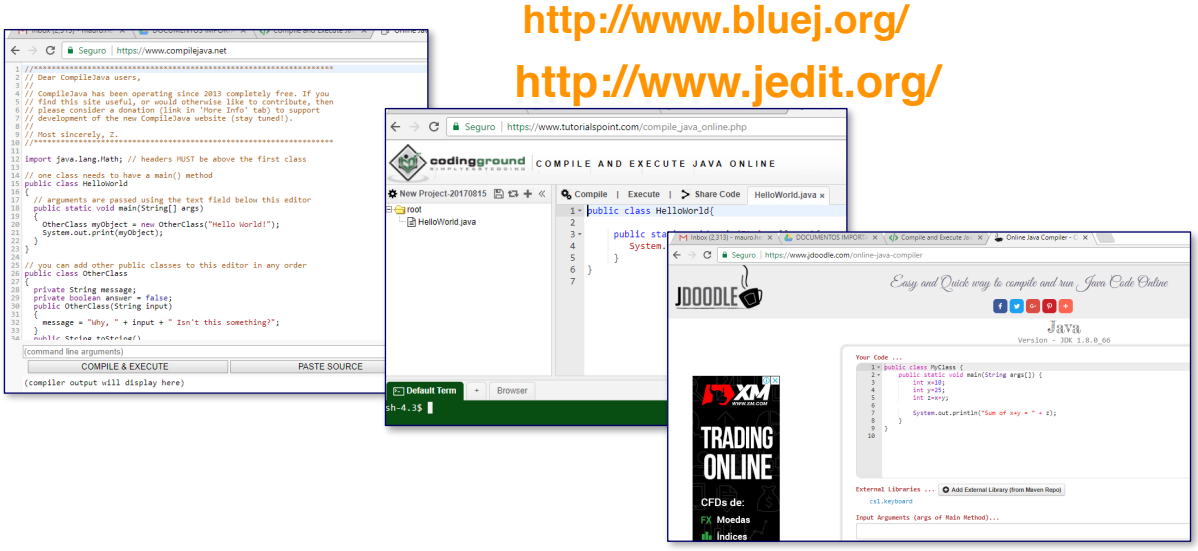
https://www.jetbrains.com/idea/features/

https://eclipse.org/

https://netbeans.org/

http://www.bluej.org/

http://www.jedit.org/



Programação Orientada a Objetos

Objetivos

- Compreender os conceitos fundamentais do Paradigma Orientado a Objetos

- O aluno ao final do curso deverá ser capaz de:
  - **Entender os padrões da programação orientada a objetos**
  - **Utilizar e entender o conjunto de funções e comandos da linguagem de programação Java.**

Programação Orientada a Objetos

Conteúdo

1. **INTRODUÇÃO**
   - Paradigma de programação orientada a objetos
   - Origens e Características da linguagem Java
   - Ambiente de desenvolvimento e execução
   - Expressões e comandos

2. **ABSTRAÇÃO E CLASSES**
   - Conceito de abstração
   - Classes e instâncias
   - Encapsulamento

3. **CLASSES EM DETALHES**
   - Relacionamentos entre Classes
   - Construtores
   - Sobrecarga
   - Atributos e métodos de classe
   - Auto-referência
   - Modularização

Programação Orientada a Objetos

Conteúdo

4. HERANÇA

Hierarquia de classes
Classes abstratas
Polimorfismo

5. EXCEÇÕES EM DETALHES

Gerando exceções
Criando exceções

Programação Orientada a Objetos

Bibliografia

1. Barnes, D.J., **Programação Orientada a Objetos com Java**, Pearson Education, 2004.

Bibliografia

1. Deitel, H.M. and Deitel, P.J., **Java Como Programar**, Editora Bookman, 2005.

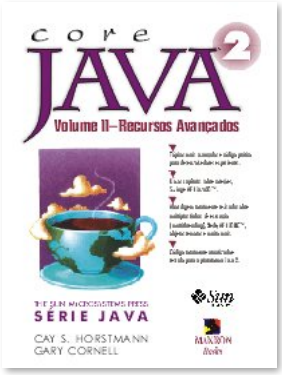
Bibliografia

1. Camarao, C., **Programaçao de Computadores em Java,** Editora LTC, 2001.

Programação Orientada a Objetos

Bibliografia

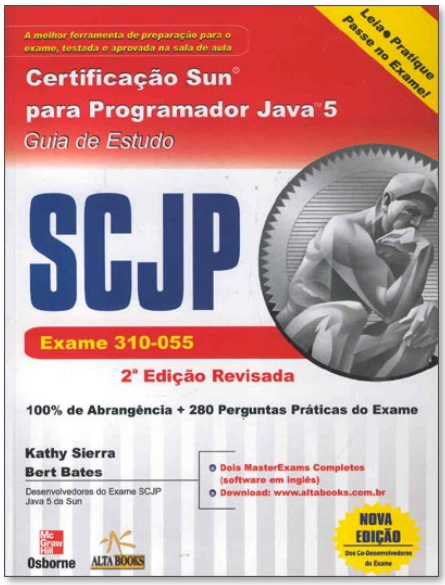
1. Horstmann, Cay S., **Core Java 2**, Pearson Education, 2001.



Programação Orientada a Objetos

Bibliografia

1. Sierra, K., **Certificação Sun para Programador Java – Guia de Estudo**.



Programação Orientada a Objetos

Avaliação

- **2 Avaliações Parciais** (P1 e P2)
  - ✓ **P1 = 20 pontos P2 = 30 pontos**
- **1 Avaliação Semetral** (S)
  - ✓ **S = 30 pontos**
- **Trabalhos e Listas de Exercícios** (T)
  - ✓ **T = 20 pontos**

Média Final (MF) = P1 + P2 + S + T

✓ Aprovação: **MF ≥ 60** Reprovação: **MF < 40**

Exame Final (EF): 40 ≤ MF ≤ 59

Nota Final (NF): (MF + EF) / 2 Aprovação: **NF ≥ 60**

Programação Orientada a Objetos

1. INTRODUÇÃO
   - 1.1. Paradigma de programação orientada a objetos
   - 1.2. Origens e Características da linguagem
   - 1.3. Ambiente de desenvolvimento e execução
   - 1.4. Expressões e comandos

Programação Orientada a Objetos

Introdução

1. Java é Linguagem e Plataforma de desenvolvimento de software
2. http://www.oracle.com/technetwork/java/index.html(http://java.sun.com)

Programação Orientada a Objetos

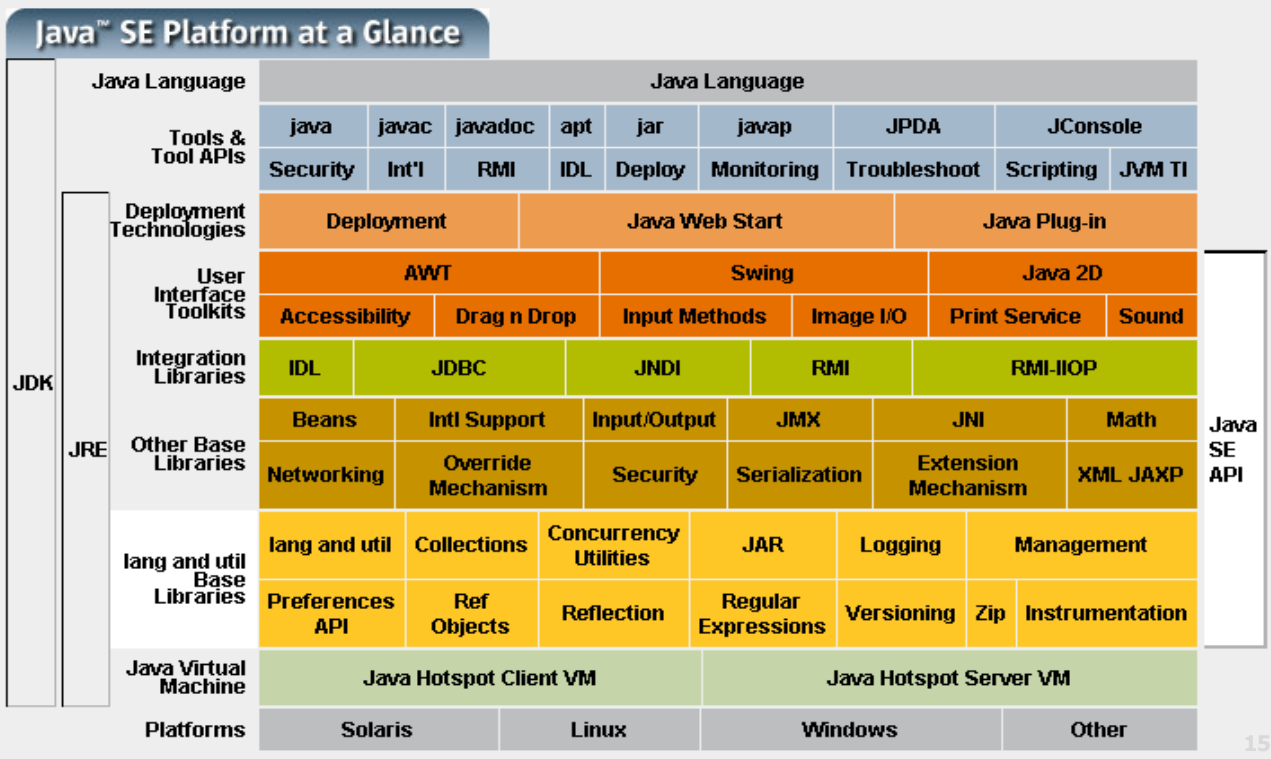
Introdução

Version	Release date	End of Free Public Updates[5][6]	Extended Support Until
JDK Beta	1995	?	?
JDK 1.0	January 1996	?	?
JDK 1.1	February 1997	?	?
J2SE 1.2	December 1998	?	?
J2SE 1.3	May 2000	?	?
J2SE 1.4	February 2002	October 2008	February 2013
J2SE 5.0	September 2004	November 2009	April 2015
Java SE 6	December 2006	April 2013	December 2018
Java SE 7	July 2011	April 2015	July 2022
Java SE 8 (LTS)	March 2014	January 2019 for Oracle (commercial) December 2020 for Oracle (non-commercial) At Least September 2023 for AdoptOpenJDK	March 2025
Java SE 9	September 2017	March 2018	N/A
Java SE 10 (18.3)	March 2018	September 2018	N/A
Java SE 11 (18.9 LTS)	September 2018	N/A for Oracle At Least September 2022 for AdoptOpenJDK	Vendor specific
Java SE 12 (19.3)	March 2019	N/A for Oracle September 2019 for OpenJDK	N/A

Legend: Old version | Older version, still supported | **Latest version** | Future release

James Gosling trabalhou desde 1984 na Sun** até abril de 2010, quando se demitiu (02/04/2010).

Faculdade Pitágoras de Uberlândia
Plataforma Java SE



Programação Orientada a Objetos
Ambiente de Desenvolvimento

1. Eclipse IDE (www.eclipse.org) – IBM (Maior projeto opensource do mundo)
2. NetBeans (www.netbeans.org) – SUN
3. Sun Studio Creator e Sun Studio Enterprise (www.java.sun.com) - SUN
4. JDeveloper (www.oracle.com) – ORACLE
5. IntelliJ (www.jetbrains.com)
6. JBuilder (www.codegear.com)
7. EditPlus
8. NotePad++
9. Etc

Programação Orientada a Objetos

Compilação

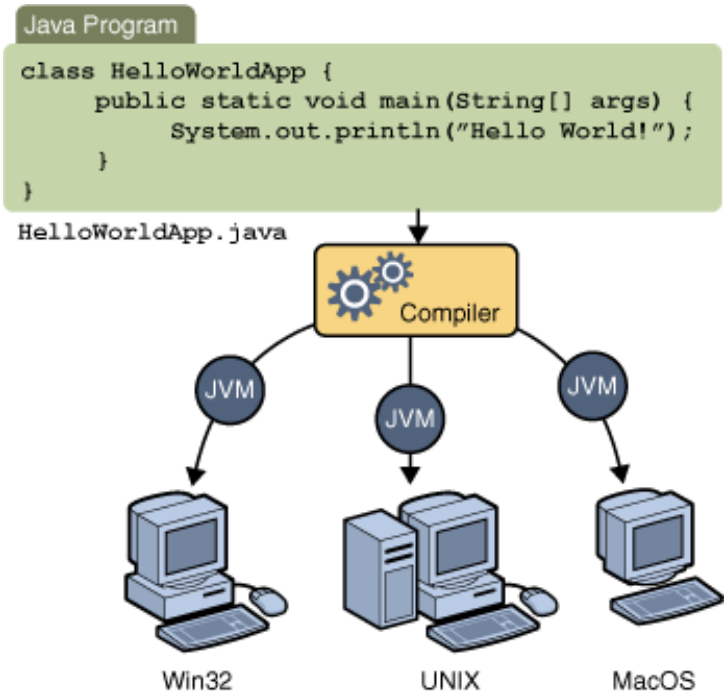
1. Todo código Java é escrito em arquivo texto.
2. Um compilador compila os fontes gerando arquivos de bytecodes (*.class)
3. A execução do programa necessita de uma instância de uma JVM na plataforma (S.O e hardware) local que interpreta os bytecodes.
4. O nome bytecode refere-se ao fato de que cada comando da JVM tem código de operação (OPCODE) de um byte
5. Veja detalhes em **http://homepages.inf.ed.ac.uk/kwxm/JVM/codeByNo.html**

Programação Orientada a Objetos

A plataforma Java

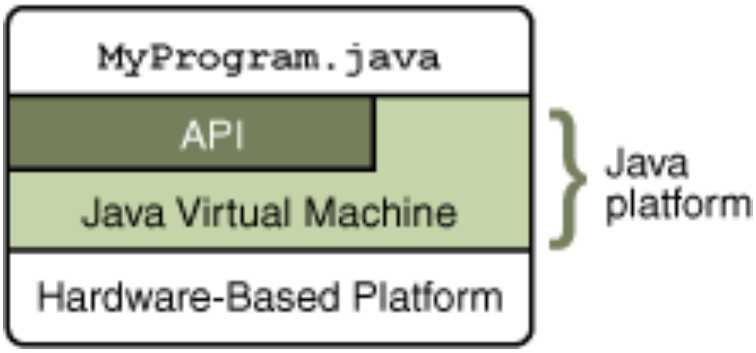
1. Programas **Java** são executados (interpretados) por outro programa chamado **Java VM**. O programa **Java** é interpretado pela **Java VM** para o S.O. nativo. Isto significa que qualquer computador com a **Java VM** instalada pode rodar programas **Java**, não importando o computador no qual a aplicação foi originalmente desenvolvida.
2. Por exemplo, um programa **Java** desenvolvido em um PC com Windows NT rodará sem modificações em uma estação Sun Ultra workstation com S.O. Solaris, e vice-versa.

Portabilidade



A plataforma Java

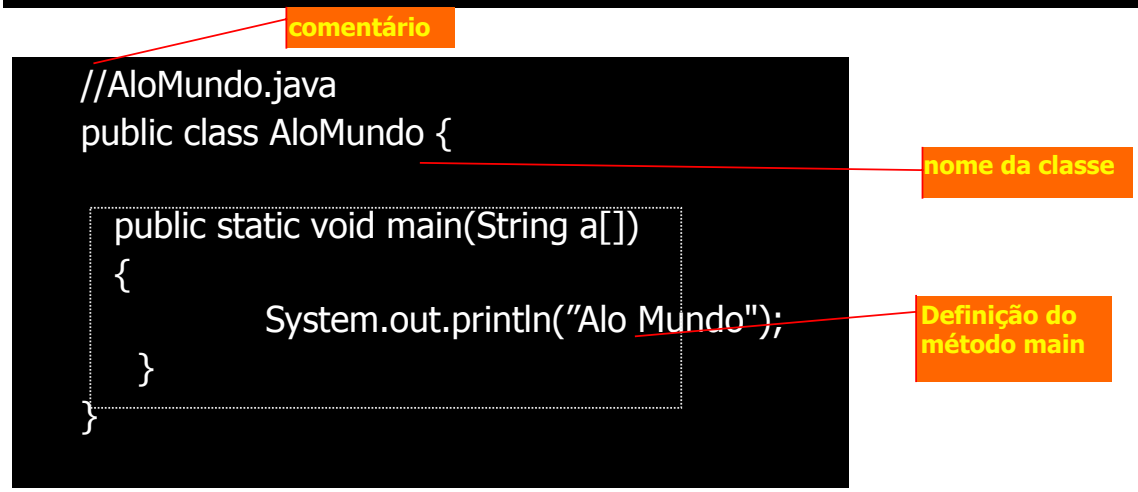
1. A plataforma Java consiste de interfaces para programação de aplicações Java (application programming interfaces – **API**) e a máquina virtual **Java** (Java virtual machine - **JVM**).
   1. **Java APIs** são bibliotecas de código compilado que você pode usar em seus programas



Programação Orientada a Objetos
Distribuições

1. **JRE** (Java Runtime Environment)
    1. JVM e APIs
2. **JDK** (Java Development Kit)
    1. JVM e APIs
    2. Ferramentas de desenvolvimento (linha de comando) – compiladores, debugadores, etc
    3. Fontes
3. O download do instalador do **JDK** ou do **JRE** pode ser obtido em http://www.oracle.com/technetwork/java/index.html
4. O instalador da **JDK** vem também com o instalador da **JRE**.

Programação Orientada a Objetos
Uma aplicação java simples



3. Para compilar (no prompt de comandos):
    1. c:/>meus_programas/javac AloMundo.java

Programação Orientada a Objetos
Uma aplicação java simples

1. O compilador java gera o arquivo AloMundo.class, que é o programa compilado para bytecodes, a linguagem da máquina Java.
2. Para executar o programa (interpretação) basta digitar no prompt de comandos:

```
c:/>meus_programas/java AloMundo
```

3. E o resultado será:

```
c:/>meus_programas/java AloMundo
Alo Mundo
```


Programação Orientada a Objetos
Dissecando o código

1. Comentários dentro do código
    1. // resto da linha é comentário
        1. Comentário é ignorado pelo compilador
        2. Documenta código
    2. /* múltiplas linhas */
    3. /* comentário de muitas
    4. linhas. */

```
public class AloMundo
```

2. Começa definição da classe AloMundo
    1. todo programa Java tem pelo menos uma classe definida pelo programador

Programação Orientada a Objetos
Dissecando o código

1. Nome da classe é um identificador
    1. Sequência de Caracteres consistindo de letras, dígitos, underscores (_) e dollar ($)
    2. Não pode começar com um dígito, e não pode conter espaços
    3. Case sensitive
        1. a1 e A1 são diferentes

3. Palavra reservada **public**
    1. modificador de acesso, torna a classe, método, variável ou objeto acessível para todos

Programação Orientada a Objetos
Dissecando o código

1. Arquivo do código-fonte
    1. Nome do arquivo é o nome da classe com extensão .java
    2. AloMundo.java
2. Chave esquerda e direita { ... }
    1. Contém a definição da classe

3. `public static void main(String a[])`

    2. Toda aplicação começa a execução pelo método main
        1. Parenteses indica que main é um método
        2. Aplicações Java contém um ou mais métodos
        3. Apenas um método pode ter o nome main

Programação Orientada a Objetos
Dissecando o código

```
{
        System.out.println(”Alo Mundo");
}
```

2. System.out
    1. Objeto de saída padrão
    2. A saída é a janela do prompt de comandos
3. Método System.out.println
    1. Imprime texto
    2. Toda instrução termina com ;
4. A definição (ou corpo) do método fica entre { ... }

Programação Orientada a Objetos
Dissecando o código

1. Caracteres de escape
    1. barra invertida (\ )
        1. \n - nova linha
        2. \r - retorno de carro
        3. \" - aspas duplas
        4. \t - tabulação
        5. \\ - barra invertida
2. Uso
    1. System.out.println(”Bem vindo\na\nJava!");
    2. Saída

```
Bem vindo
a
Java!
```

Objetos e Java

Definindo classes Java

Objetos

1. **Objeto** em software é uma maneira de representar as coisas do mundo real.
2. Objeto é um modelo abstrato das **coisas** (reais ou virtuais) do mundo real.
3. Coisas tais como um cliente ou agenda de telefones ou uma folha de pagamento ou uma tela com um formulário ou até um simples botão de uma interface gráfica com o usuário.

Programação Orientada a Objetos

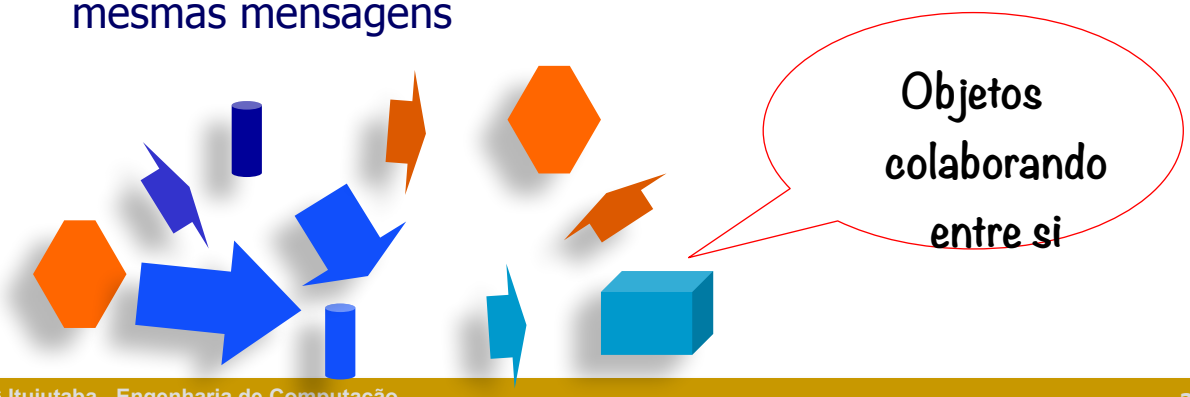
Objetos

1. Objetos possuem:
   1. **Identidade**
      1. Cada objeto tem sua própria existência, ou seja, eles "vivem" na memória do computador.
   2. **Estado**
      1. Conjunto de características com valores definidos
   3. **Comportamento**
      1. Como eles reagem ao mundo. Também dizemos que são as "mensagens" que eles recebem. Geralmente, um comportamento altera o estado do objeto.

Programação Orientada a Objetos

Programa Orientado a Objetos

1. Características de um **programa** OO
   1. Tudo é **objeto**
   2. Um programa é uma **coleção** de objetos colaborando entre si através do envio de mensagens uns aos outros
   3. Todo objeto possui um tipo (que descreve seus dados)
   4. Objetos de um determinado tipo podem receber as mesmas mensagens



Programação Orientada a Objetos
Tipos primitivos e tipo de objeto

1. Em **Java** tudo é objeto, exceto alguns valores "primitivos"
   1. uma janela é objeto, um botão de uma interface gráfica com o usuário é um objeto, uma conexão com um banco de dados é um objeto, um programa é um objeto, uma palavra é um objeto, ou seja, quase tudo exceto os primitivos.
2. Tipos primitivos
   1. Inteiros: byte, short, int, long
   2. Reais: float, double
   3. Caracter: char
   4. Lógico: boolean

Programação Orientada a Objetos
Exemplos de tipos primitivos e literais

1. Literais de caracter
   1. char c = 'a';
   2. char z = '\u0041'; // em Unicode
2. Literais inteiros
   1. int i = 10; short s = 15; byte b = 1;
   2. long hexa = 0x9af0L; int octal = 0633;
3. Literais de ponto-flutuante
   1. float f = 123.0f;
   2. double d = 12.3;
   3. double g = .1e-23;

Programação Orientada a Objetos
Exemplos de tipos primitivos e literais

1. Literais booleanos
    1. boolean v = true;
    2. boolean f = false;
2. Literais de string (não é tipo primitivo - s é uma referência)
    1. String s = "abcde";
3. Literais de vetor (não é tipo primitivo - v é uma referência)
    1. int[] v = {5, 6};

Programação Orientada a Objetos
Objetos

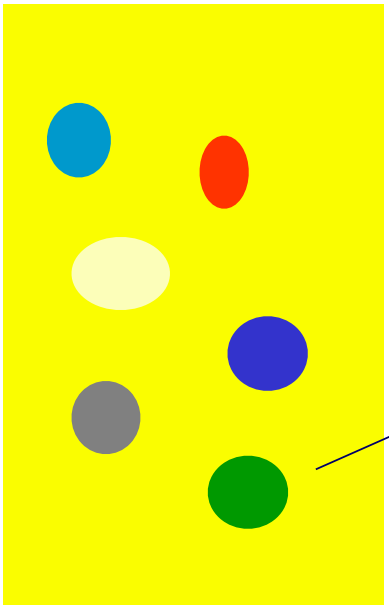
1. Em **Java**, objetos são armazenados na memória e manipulados por meio de uma referência

3. Os objetos possuem valores que definem suas características (estado), funções que definem seu comportamento (métodos) e identidade (referência)

5. Um programa **Java** não manipula diretamente seus objetos. Deve-se usar referencias aos objetos para usá-los

7. No livro **"Thinking in Java"** de **Bruce Eckel** ele faz uma analogia entre **objetos** e sua **referência** e uma TV e seu controle remoto (veja ilustração na próxima transparência)

Objeto e sua referência



memória RAM do computador

As variáveis em programas Java podem armazenar dois tipos de valores:

x = 1;

x = new objeto();

1) Valores primit

2) R cias a

- A tv é o objeto e o controle remoto é a referência

O que é classe?

1. **Classe** é um documento que descreve um tipo de objeto
2. Uma classe representa um tipo de dados complexo
3. Classes descrevem
   1. **Tipos** dos dados que compõem o objeto (o que podem armazenar)
   2. **Procedimentos** que o objeto pode executar (o que podem fazer)

Programação Orientada a Objetos
Construção de Classes

1. **Declaração**

```
[modificadores] class NomeClasse
                                [extends SuperClass
                                [implements Interfac
{
            atributos
            métodos
}
```

4. **Modificadores**
   1. **Classe pública** (**public**): a classe pode ser utilizada por objetos de fora do pacote. Por default, a classe só pode ser acessada no próprio pacote
   2. **Classe Abstrata** (**abstract**): não pode ter objetos instanciados
   3. **Classe final** (**final**): a classe não pode ter subclasses

Programação Orientada a Objetos
Atributos da Classe

1. **Declaração de Atributos**
   1. [modificador] [chaves] tipo nomeAtributo [= expressão];
2. **Modificador**
   1. **public**: o mundo inteiro pode acessar
   2. **protected**: somente os métodos da classe e de suas subclasses podem acessar, ou ainda, estando na mesma package
   3. **private**: somente os métodos da classe podem acessar o atributo
3. **Chaves**
   1. **static**: o atributo é da classe, não do objeto, logo, todos os objetos da classe compartilham o mesmo valor deste atributo
   2. **final**: o valor do atributo não pode ser alterado (constante)
   3. **transient**: o atributo não é serializado (não é persistente)

1. Declaração de Métodos

```
[modificador] [chaves] tipoRetorno nomeMétodo (
  [parâmetros] ) [throws exceptions]
{
  corpo do método (lógica)
}
```


1. chaves:
   1. **static**: método da classe e não das instâncias
   2. **abstract**: utilizado somente em classes abstratas, o método não tem corpo
   3. **final**: o método não pode ser sobre-escrito
   4. **synchronized**: declara o método como zona de exclusão mútua no caso de programas concorrentes
2. a passagem de parâmetros em Java é sempre por valor.
3. Um método é identificado pelo seu nome e pelos parâmetros **(assinatura do método)**

Programação Orientada a Objetos
Inicialização de Objetos

1.Construtor da Classe

1. tem o mesmo nome que a classe
2. é chamado na criação do objeto **(new)**

```
class Teste {
      public Teste() { .... }
      public Teste(int i) { .....    }
      .....
}
```


Programação Orientada a Objetos
Exemplo

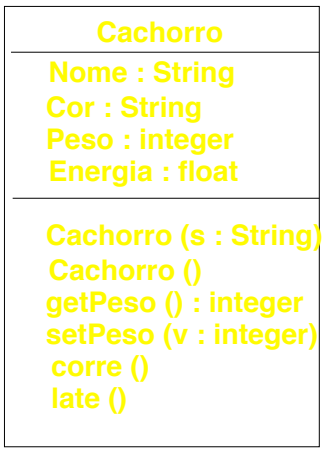
```
class Cachorro {

   // Atributos dos objetos da classe
   private String nome;
   private String cor;
   private int    peso;
   private float  energia;

   // Construtores
   Cachorro(String s) { nome = s; }
   Cachorro() { nome = "Sem nome"; }

   // Métodos (comportamentos dos objetos da classe)
   void setPeso(int v) { peso = v; }
   int getPeso() { return peso; }

   void corre() { ... }
   void late() { ... }
}
```



Programação Orientada a Objetos

Exercícios

1. 1. Construa as seguintes classes:

2. Uma Pessoa tem um nome (String)
3. Uma Porta tem um estado aberto, que pode ser true ou false, e pode ser aberta ou fechada
4. Uma Casa tem um proprietário Pessoa e um endereço
5. Um Ponto tem coordenadas x e y inteiras
6. Um Circulo tem um Ponto e um raio inteiro
7. Um Pixel é um tipo de Ponto que possui uma cor

Programação Orientada a Objetos

Exercícios

1. 2. Escreva uma classe Ponto
   1. contém x e y que podem ser definidos em construtor
   2. métodos getX() e getY() que retornam x e y
   3. métodos setX(int) e setY(int) que mudam x e y
2. 3. Escreva uma classe Circulo, que contenha
   1. raio inteiro e origem Ponto
   2. construtor que define origem e raio
   3. método que retorna a área
   4. método que retorna a circunferência
   5. use java.lang.Math.PI (Math.PI)
3. 4. Crie um segundo construtor para Circulo que aceite
   1. um raio do tipo int e coordenadas x e y