

Revisão

- Conceitos de lógica de programação
 - if-else
 - for | while | do-while
 -

- **casting**

```
short s = 120;
```

```
byte b = s; // erro porque o tamanho do byte é menor  
que short
```

```
byte b = (byte) s;
```

Revisão

- Promoção automática
 - Operações entre **bytes**, os tipos são promovidos para **int**

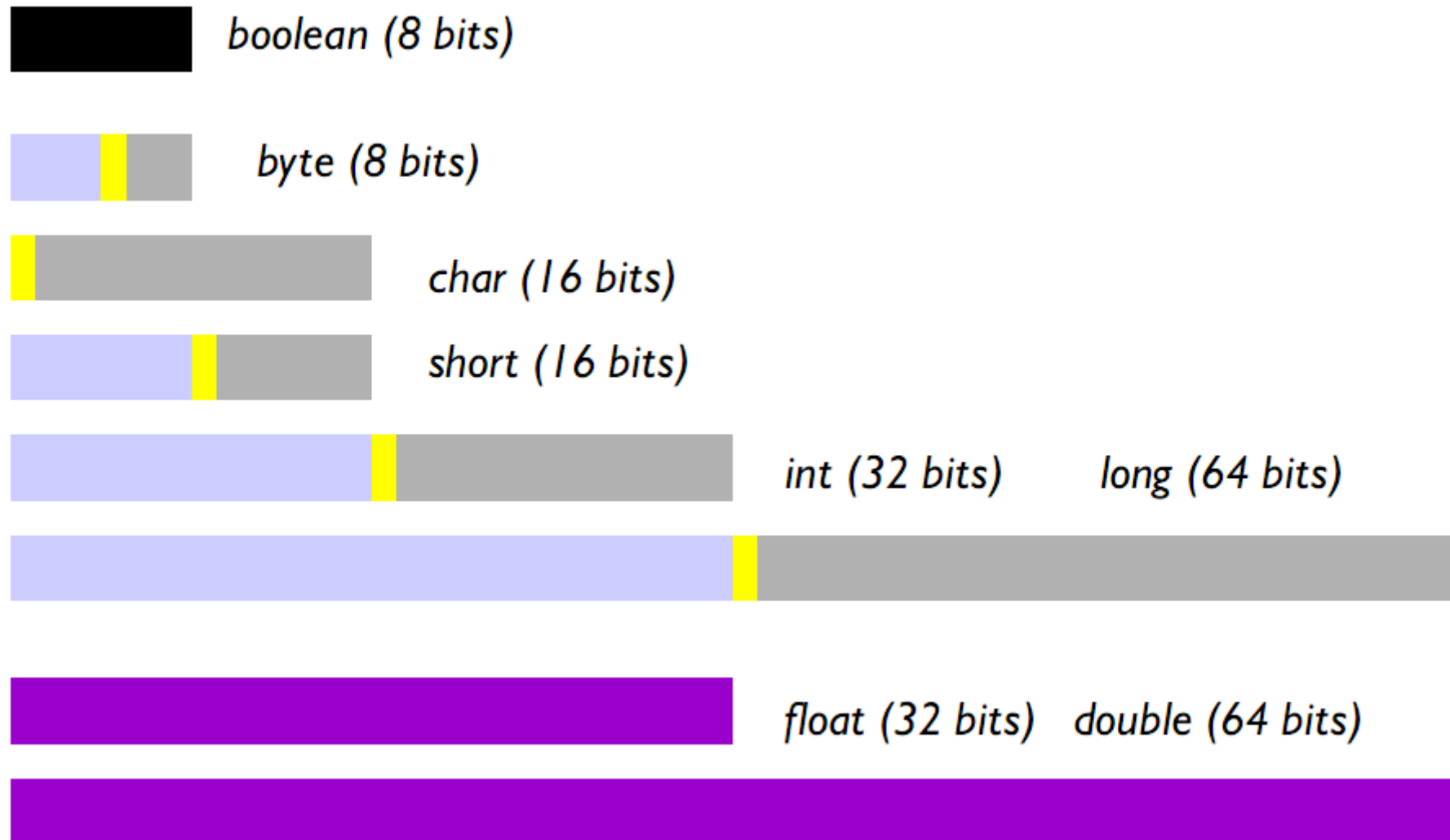
```
byte a = 10;
```

```
byte b = 23;
```

```
byte c = a + b // erro! precisa de um cast
```

```
byte c =(byte) (a+b)
```

Revisão



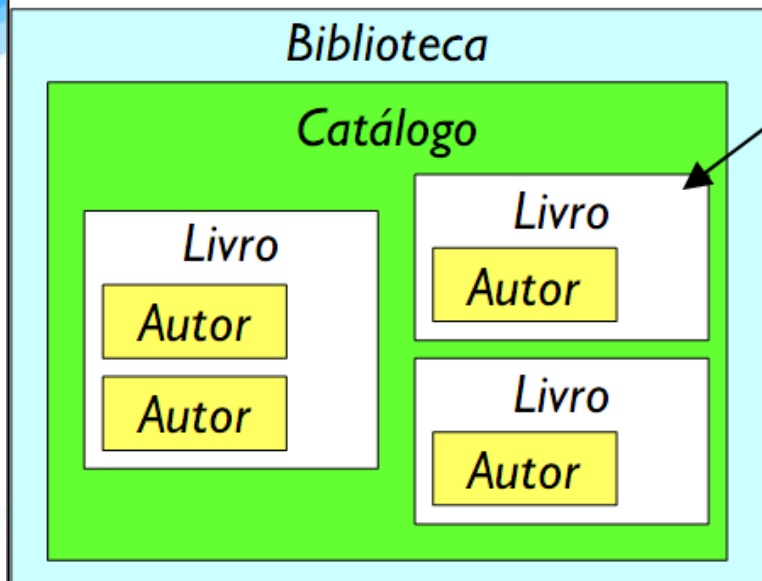
Orientação a Objetos

- Paradigma de desenvolvimento
 - Análise, projeto (design) e programação
- O que o sistema deve fazer? Quais as atividades de cada 'ator' do sistema?
- sistema é decomposto em diversos objetos com diferentes características.
 - Abstrações mais simples e próximos do mundo real

Orientação a Objetos

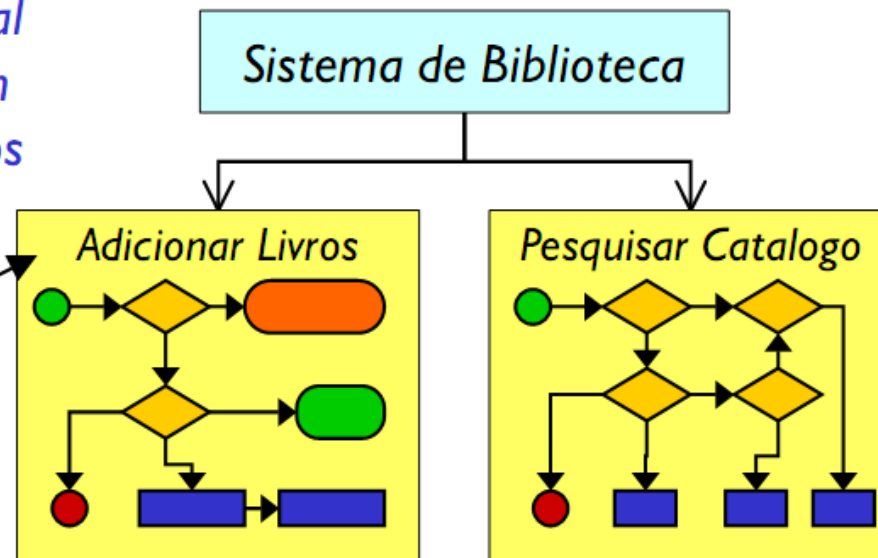
- Um programa desenvolvido com uma linguagem de programação orientada a objetos manipula estruturas de dados através dos objetos da mesma forma que um programa em linguagem tradicional utiliza variáveis.
- A análise e projeto orientados a objetos têm como meta identificar o melhor conjunto de objetos para descrever um sistema de software. O funcionamento deste sistema se dá através do **relacionamento e troca de mensagens entre estes objetos.**

Orientação a Objetos



Lógica procedural encapsulada em objetos pequenos

Lógica exposta e espalhada por todo o sistema



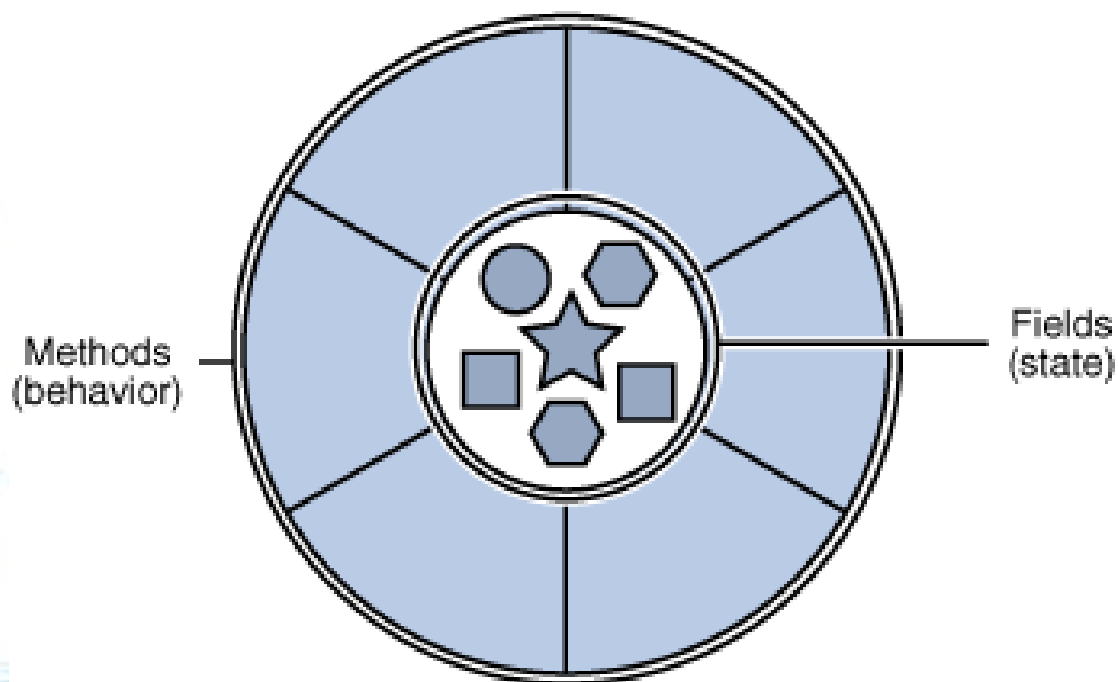


Orientação a Objetos

- Então no paradigma orientado a objeto, os sistemas são compostos por objetos e relacionamento entre eles.

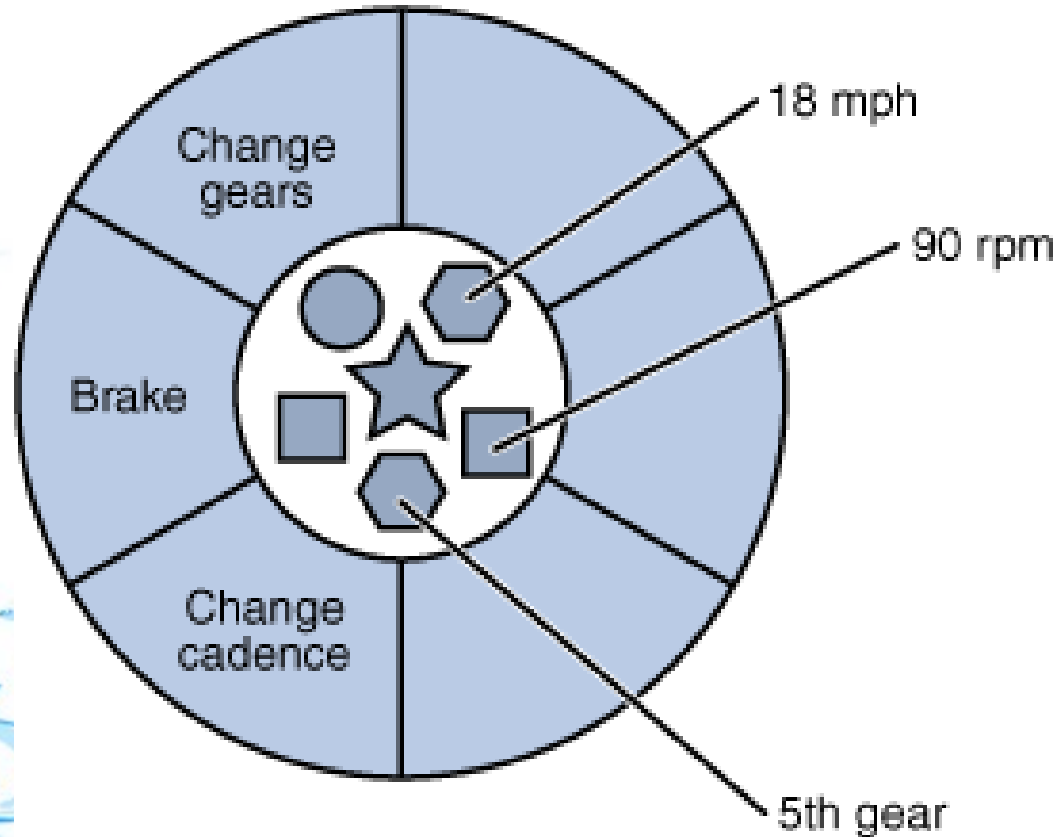
Orientação a Objetos

- Objeto
 - composto por estado e comportamento



Orientação a Objetos

- Objeto



Orientação a Objetos

- Objeto
 - Objetos são armazenados na memória heap e possuem suas referências no stack(pilha), o que?!?

Orientação a Objetos

- Tipos primitivos não são objetos em Java e por isso são armazenados na pilha.
- Objeto
 - Estado (variáveis)
 - Comportamento (métodos)
 - Identidade (referência)

Orientação a Objetos

- Algumas observações:
 - Java não possui aritmética de ponteiros
 - Endereços de memória (stack) são inacessíveis ao programador!

Orientação a Objetos

- Voltando aos operadores...

Object a = new Object();

Object b = new Object();

– **If (a == b) ?**

– **a=b; ?**

Orientação a Objetos

- **Classes**
 - Muitos objetos possuem características comuns e pertencem a uma classe de objetos.
 - Mamíferos – humanos

Orientação a Objetos

- **Classes**

- Uma classe é uma espécie de 'planta baixa'. Os objetos são construídos a partir de informações desta planta.

- Especificação do objeto

- Tipos de dados que compõe o objeto

- Procedimentos(comportamento) de um objeto

Orientação a Objetos

- Um objeto é uma instância de uma Classe, ou seja, ele possui um estado específico que o diferencia dos outros objetos que sejam da mesma classe...

```
Object a = new Object();
```

```
Object b = new Object();
```

a e b pertencem a mesma Classe (Object) mas possuem identidades diferentes e estados diferentes.

Orientação a Objetos

- Os relacionamentos entre as classes é estático, ou seja definido no código enquanto os objetos podem ser criados e destruídos durante a execução do programa.
- Durante a execução do programa não existem classes, apenas objetos!
- A classe define representa vários objetos que ocupam espaço na memória

Orientação a Objetos

- Como definir classes ?

```
Class NomeDaClasse {  
    escopo da classe  
}
```

Orientação a Objetos

- Um arquivo .java pode ter várias classes mas o nome do arquivo será igual ao nome da classe definida como **public**

```
class UnB1 {  
    Escopo da classe 1  
}  
  
public class Aula2 {  
    Escopo da classe 2  
}
```

Orientação a Objetos

- Como vimos as Classes possuem atributos e comportamentos ... (variáveis e métodos)
- Escreva a Classe Humano

```
class Humano {  
    int idade;  
    int altura;  
    ...  
}
```

Orientação a Objetos

- Métodos
 - Métodos representam os comportamentos ou ações que uma classe pode ter
 - Andar, falar, comer, ouvir ...

Orientação a Objetos

- Métodos
 - Os métodos podem ou não ter argumentos e podem ou não ter retornos

boolean andar(int passos);

int comer(String comida);

void ouvir();

void falar(String afirmacao);

Orientação a Objetos

Vamos agora incrementar a Classe Humano

```
class Humano {
```

```
    int idade;
```

```
    int altura;
```

```
    boolean andar(int passos);
```

```
    int comer(String comida);
```

```
    void ouvir(Humano pessoa); // void ouvir();
```

```
    void falar(String afirmacao);
```

```
    ...
```

```
}
```

Orientação a Objetos

- Uma vez construída a Classe, ela é instanciada na forma de um objeto, ou seja, uma vez terminada a planta basta agora construir a Casa.

```
Humano pessoa1 = new Humano();
```

```
pessoa1.idade = 15;
```

```
pessoa1.andar(3);
```

Orientação a Objetos

- Assim como nós dizemos construir uma casa, também podemos falar na construção de um objeto a partir da definição da sua Classe...
- Mas os construtores de uma casa deixam ela no seu estado inicial, então devem existir construtores de objetos que deixam eles no seus estados iniciais ...

Orientação a Objetos

- Métodos Construtores
- Humano pessoa1 = new **Humano()**;
- O método Humano() é o construtor da classe Humano...

Orientação a Objetos

- Voltando a classe Humano

```
Class Humano {
```

```
...
```

```
    public Humano(){
```

```
        Idade = 15;
```

```
        Peso = 70;
```

```
        ...
```

```
    }
```

```
}
```

Orientação a Objetos

- No momento da instanciação da Classe o construtor é invocado colocando o objeto no seu estado inicial...
- Caso não haja construtor, o compilador coloca um construtor vazio (sem nada)

Orientação a Objetos

- Construtores ...
 - Mas podem existir diferentes construtores numa casa, pra alguns você vai ter de falar o que fazer para outros eles já vão saber...

Orientação a Objetos

- Um objeto pode também ter múltiplos construtores com argumentos diferentes..

```
public Humano();
```

```
public Humano(int idade, int peso);
```

```
public Humano(int idade);
```

Orientação a Objetos

```
Humano pessoa1 = new Humano();
```

```
Humano pessoa2 = new Humano(15,70);
```

```
Humano pessoa3 = new Humano(23);
```

```
pessoa1.andar(3);
```

```
pessoa2.ouvir(pessoa1);
```

```
pessoa3.comer("salgado");
```

Orientação a Objetos

- Como comparar objetos no heap?
 - ==** (comparam a identidade no stack)
 - Deve ser feito através de algum **comportamento**, onde um objeto se compara com outro objeto...
 - Comportamento = método!

Orientação a Objetos

- Crie um método de comparação entre 2 humanos...

```
boolean isEqual(Humano pessoa2)
```

Orientação a Objetos

- Herança (introdução)
 - Um Humano é um mamífero, logo os humanos herdam características que fazem parte de todos os mamíferos...
 - Da mesma forma uma classe de objetos pode pertencer a um grupo maior de outros objetos..

Orientação a Objetos

- Herança (introdução)
 - Veículo > Carro
 - Veículo > Bicicleta
 - Mamífero > Humano > Programador
 - Mamífero > Humano > Engenheiro

Orientação a Objetos

- Herança (introdução)
 - A palavra usada em Java para expressar herança é “extends”
 - class Humano **extends** Mamifero

Orientação a Objetos

- O que a herança faz?
 - Ela herda 'todos' os comportamentos(**métodos**) instintivos da classe mãe, ou classe “**super**”
 - E também as características (**variáveis**)
 - A herança gera a especialização das Classes