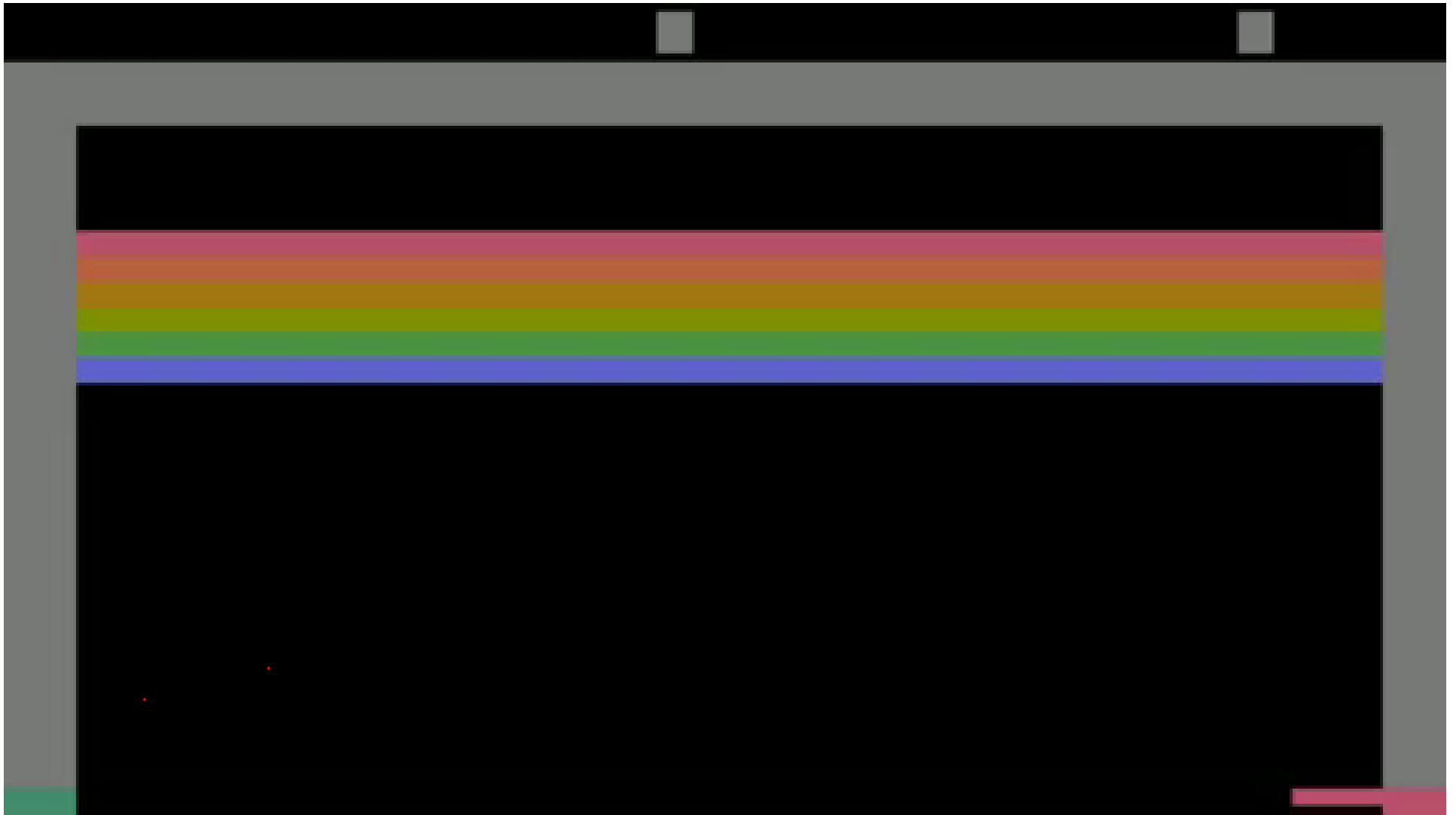


Computação II – Orientação a Objetos

Fabio Mascarenhas - 2014.1

<http://www.dcc.ufrj.br/~fabiom/java>

Breakout



Componentes do Breakout

- Bola
- "Raquete"
- Tijolos
- Paredes *→ campos na classe Jogo*
- Score *e vidas → campos na classe Jogo*
- Nem todos vão precisar de classes próprias para representa-los!

Métodos

- Um *método* é uma operação de um objeto
- Sintaticamente, um método se parece com uma função: é declarado dentro de uma classe (mas sem o palavra-chave `static`), possui uma lista de parâmetros e um bloco de comandos, e pode retornar valores
- Mas, para chamar um método, precisamos dizer qual objeto vai *receber* a chamada: `<obj_rec>.método(<arg1>, ..., <argn>)`
 - `"ab".equals("a" + "b"), scan.nextInt()`
- Dentro de um método, o variável `this` é o objeto que está recebendo a chamada, e podemos ter acesso aos campos e métodos desse objeto

Inicialização fora do construtor

- Às vezes queremos que todo campo de um objeto seja inicializado da mesma forma, independente do que foi passado para o construtor
- Podemos fazer a inicialização direto na declaração do campo:
 - `int tamanho = 30;`
- Todo objeto criado vai começar com tamanho igual a 30, mas depois o valor de tamanho de cada um pode divergir

Inicialização fora do construtor, cont.

- A inicialização é feita sempre que um novo objeto for criado, então cada objeto pode receber um valor diferente:
 - `public Cor cor = new Cor(Math.random(), Math.random(), Math.random());`
- Cada objeto criado vai receber **uma nova cor** em que cada componente é determinado aleatoriamente
- É como se a inicialização estivesse sendo feita dentro de **cada** construtor da classe

Campos da classe (*campos estáticos*)

- E se queremos que **todos** os objetos de determinada classe tenham um campo que sempre vai ter o **mesmo** valor?
- Nesse caso, usamos um *campo estático*, declarado com a palavra-chave `static`
 - `static int tamanho = 30;`
- Podemos acessar campos da classe por qualquer instância, ou diretamente pela classe
 - `Tijolo.largura`, `raquete.vx`

Campos da classe, cont.

- Para a linguagem, não há diferença entre campos estáticos e variáveis globais; a diferença está no que elas representam
- Uma variável global está associada sintaticamente a uma classe, mas é um valor que não está ligado ao comportamento de suas instâncias
 - Exemplos de cores pré-allocadas dentro da classe `Cor`
- Um campo da classe é um valor que afeta o comportamento de todas as instâncias da classe, e uma maneira de fazer elas compartilharem algum dado
 - Tamanho dos tijolos e velocidade da raquete no *breakout*

Classes parametrizadas (*classes genéricas*)

- Várias classes da biblioteca padrão de Java têm *parâmetros*
- Por exemplo, todas as classes que representam coleções (listas, mapas, conjuntos) recebem um parâmetro que diz quais são os objetos que fazem parte da coleção
- Os parâmetros são outras classes, e aparecem entre <>
 - `Set<String>`, `ArrayList<Tijolo>`, `HashMap<String, Aluno>`
- Depois veremos como definir nossas próprias classes parametrizadas

As classe HashSet e ArrayList

- Duas das coleções mais comuns usadas em programas Java são as classes HashSet e ArrayList
- Respectivamente elas representam conjuntos e listas de elementos, e são parametrizadas pelo tipo dos elementos que você quer guardar
- Use um conjunto se você não se importa com a ordem dos elementos, mas quer checar se um objeto pertence ao conjunto ou não, e não quer ter elementos duplicados
- Use uma lista se a ordem em que os elementos está importa, e você quer acessar um elemento pela posição dele na lista

Laço for de coleções

- Java tem uma versão do laço for que é especializada para percorrer uma coleção (um vetor, ou instâncias de uma das classes de coleção como HashSet e ArrayList)
- O bloco do laço é executado para cada elemento da coleção, com a variável de controle apontando para o elemento

```
for(int i: vetor) {  
    System.out.println(i);  
}
```