Computação II – Orientação a Objetos

Fabio Mascarenhas - 2014.1

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

De C para “Java com sabor de C”

```
#include  <stdio.h>

int main (void) {
    int vetor[5], i;
    int trocou = 0;
    int fim = 5;
    int temp;
    printf("Entre com um vetor de %d elementos\n", 5);
    for (i = 0; i < 5; i++) {
        printf("Elemento %d ", i);
        scanf("%d", &vetor[i]);
    }
    do {
        trocou = 0;
        for (i=0; i < fim-1; i++) {
            if (vetor[i] > vetor[i+1]) {
                temp = vetor[i];
                vetor[i] = vetor[i+1];
                vetor[i+1] = temp;
                trocou = 1;
            }
        }
        fim--;
    } while (trocou);
    for (i=0; i < 5; i++) printf("%d\n", vetor[i]);
    return 0;
}
```

Classes

- Uma das unidades básicas de um programa Java
- No nível mais simples é como uma *struct* de C, agrupando diversos valores em uma mesma entidade, e possui *campos*
- Campos de uma classe podem ter qualquer tipo, inclusive outras classes
- *Instâncias* de uma classe também são criadas com `new`

Construtor

- Podemos inicializar os campos de uma nova instância de modo parecido do que fazemos em C, mas esse não é o estilo OO apropriado
- A inicialização dos campos de um objeto é tarefa do seu *construtor*
- A declaração de um construtor se parece com a declaração de uma função em C, mas um construtor sempre tem o mesmo nome da classe
- Podemos passar argumentos para a classe quando usamos `new`, e esses argumentos são os parâmetros do construtor
- Dentro do construtor, a variável especial `this` aponta para o objeto recém-criado que deve ser inicializado

Sobrecarga de construtores

- Uma classe pode querer inicializar objetos de várias maneiras
- Para isso, podemos definir vários construtores, contanto que as *assinaturas* deles sejam diferentes (número e/ou tipos dos parâmetros)
- Essa “sobrecarga” é bastante comum nas bibliotecas Java

Identidade de objetos

- Java também tem as operações de comparação `==` (igual a) e `!=` (diferente de)
- Para números e booleanos elas dizem se dois números ou booleanos são iguais ou não
- Mas, para objetos (incluindo strings e vetores), essas operações dizem se os dois objetos que estamos comparando são *a mesma instância* ou não
- Todo objeto possui uma identidade; quando criamos um objeto com `new`, a identidade dele é diferente de todos os outros objetos, mesmo objetos da mesma classe, com o conteúdo dos campos idêntico

Identidade de objetos, cont.

- Com strings a coisa é mais sutil, às vezes dois literais idênticos acabam dividindo uma instância de `String`, às vezes não
- Se queremos comparar objetos por *conteúdo* ao invés de por identidade, precisamos usar o método `equals`
  - `"ab".equals("a" + "b") == true`
- Todo objeto tem um método equals, mas nem sempre ele funciona
  - `(new int[] { 1 }).equals(new int[] { 1 }) == false`

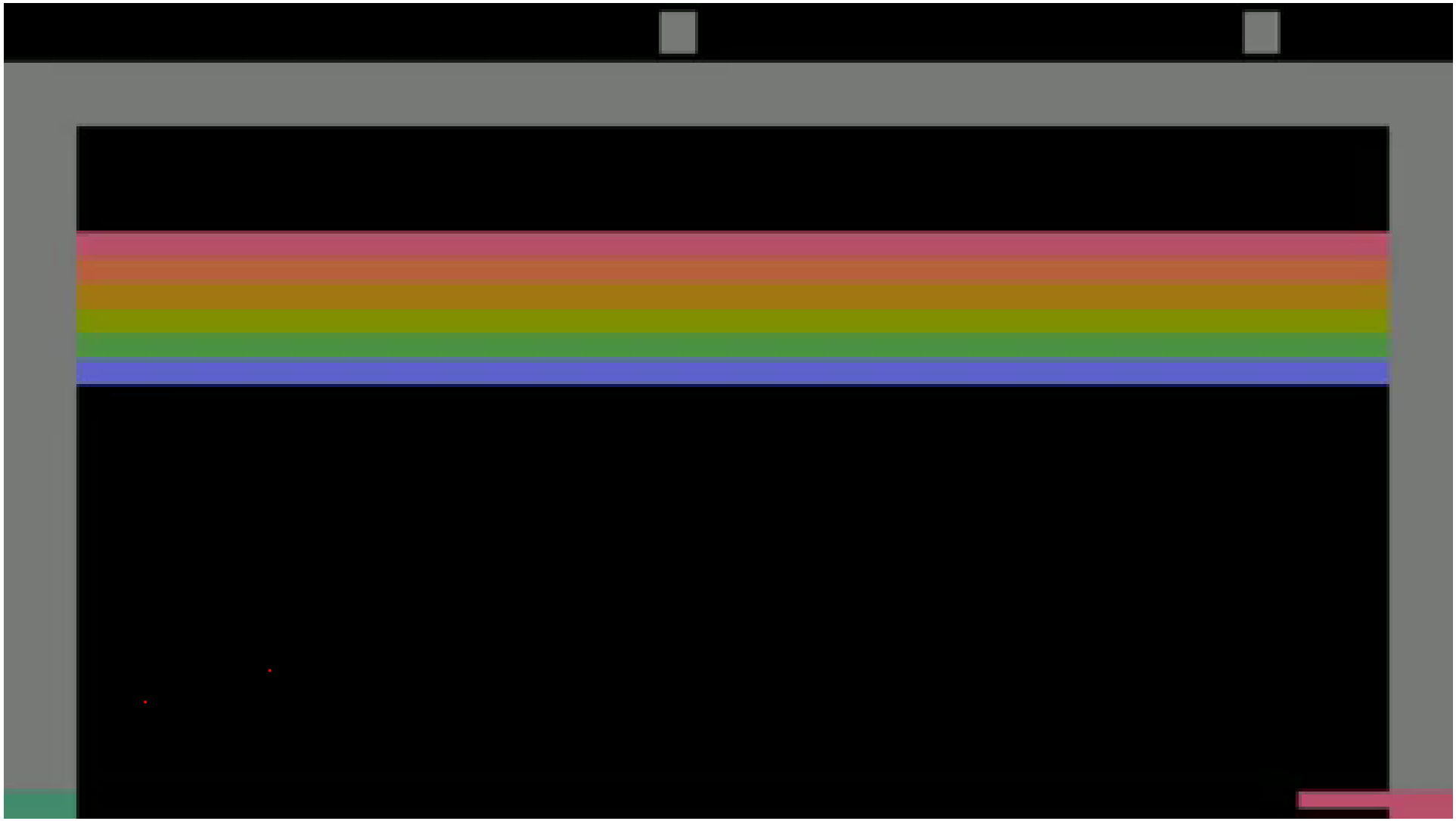
Frameworks

- É bastante comum que uma aplicação OO seja construída para usar um *framework* (arcabouço)
- Um framework é como uma máquina com algumas peças faltando, que serão fornecidas pela aplicação
- Essas “peças faltando” são instâncias de classes cuja estrutura é ditada pelas necessidades do framework
- Os objetos do framework interagem com os objetos da aplicação, que por sua vez interagem de volta com objetos do framework, para requisitar serviços

Um framework simples para jogos

- Vamos usar um framework bem simples para construir jogos 2D
- O framework fornece uma *tela* para desenhar figuras geométricas simples, além de texto
- A cada “tique” do relógio interno do framework, ele fornece uma tela em branco preta
- Ele também avisa a aplicação de *eventos* que acontecem: teclas pressionadas, e a própria passagem do tempo

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