Classes e Objetos

POO

Prof. Marcio Delamaro

POO (wikipedia)

- Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which are data structures that contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.
- A distinguishing feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated

Exemplo banal

Quero representar os dados de uma pessoa

Exemplo banal

Quero representar os dados de uma pessoa

Data nascimento, peso, altura.

Exemplo banal

Quero representar os dados de uma pessoa

Em cima desses dados, eu quero computar coisas que me interessam: idade, IMC, etc.

Data nascimento, sexo, peso, altura.

24/08/1963, M, 72, 172

Exemplo banal

Quero representar os dados de uma pessoa

Em cima desses dados, eu quero computar coisas que me interessam: idade, IMC, etc.

Data nascimento, sexo, peso, altura.

24/08/1963, M, 72, 172
idade() imc()

Exemplo banal

Quero representar os dados de uma pessoa

Data nascimento, sexo, peso, altura.

Em cima desses dados, eu quero computar coisas que me interessam: idade, IMC, etc.

24/08/1963, M, 72, 172
idade() imc()

24/10/1980, F, 51, 165
idade() imc()

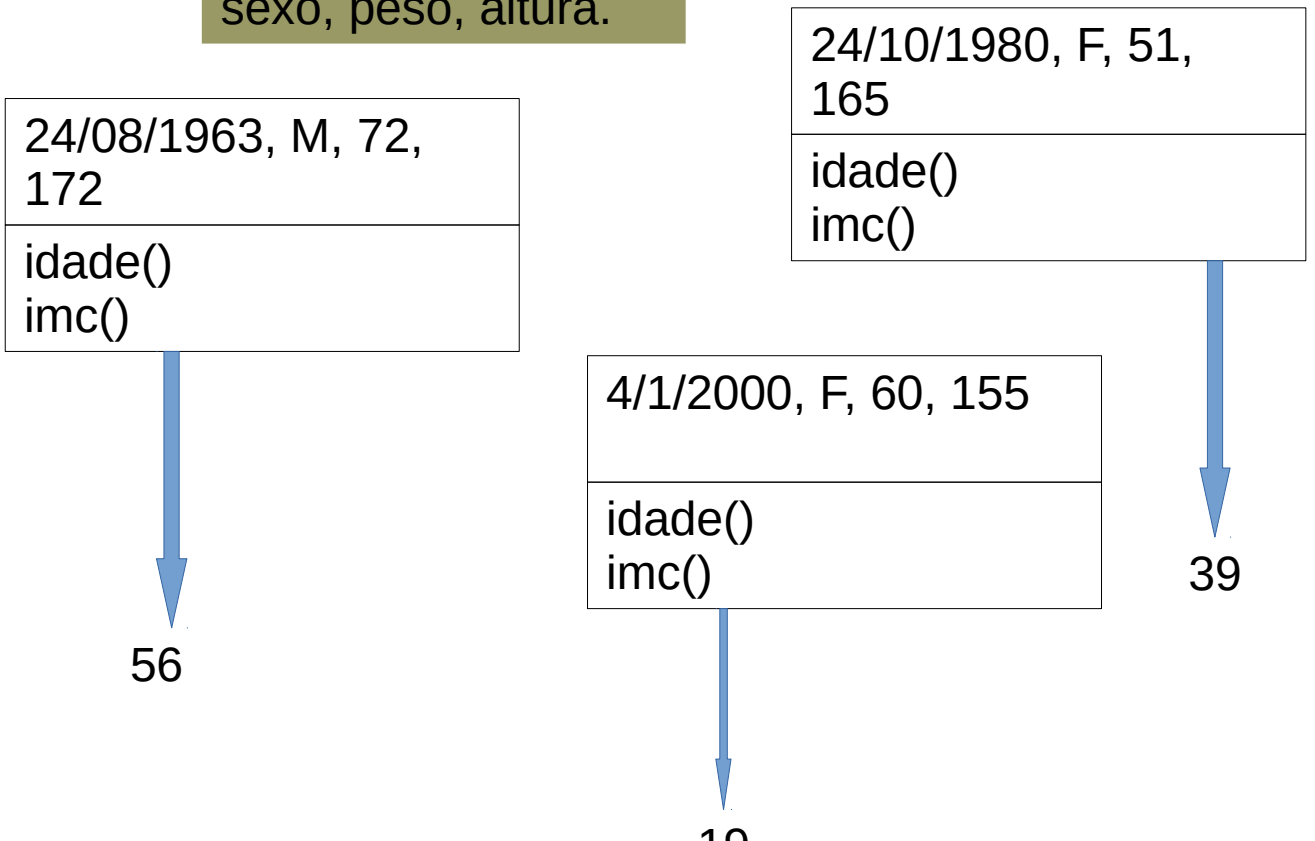
4/1/2000, F, 60, 155
idade() imc()

Exemplo banal

Quero representar os dados de uma pessoa

Data nascimento, sexo, peso, altura.

Em cima desses dados, eu quero computar coisas que me interessam: idade, IMC, etc.



Cada objeto possui os mesmos métodos (código) e dados diferentes.
Ao invocar um método esse vai considerar os dados correspondentes ao seu objeto.
Assim, os comportamentos (resultados) são diferentes para cada objeto.

Classe

- Cada objeto possui os mesmos elementos
- Atributos e métodos
- Ao definirmos (programarmos) a estrutura de um objeto, estamos definindo uma classe
- Grosso modo, em uma classe definimos um tipo de objetos, compostos por atributos (variáveis) e métodos (código).

Exemplo não tão banal

- Vamos implementar um programa para selecionar números aleatórios para a mega-sena.
- Para isso vamos precisar de uma classes que gere números aleatórios.
- Vejamos como faremos.

Geração de N.A.

- A geração começa com um inteiro x_0 chamado de semente.
- Para calcular o próximo número aleatório, fazemos $x_{i+1} = (a + m\ x_i) \bmod p$
- Vamos usar os seguintes valores: a = 453816693, m = 843314861 e p = 2147483648.
- Como semente usamos 1023.
- Queremos implementar dois métodos: `getRand()` e `getIntRand(int m)`

Atributos

- Quais seriam os atributos que compõem a minha classe?

Atributos

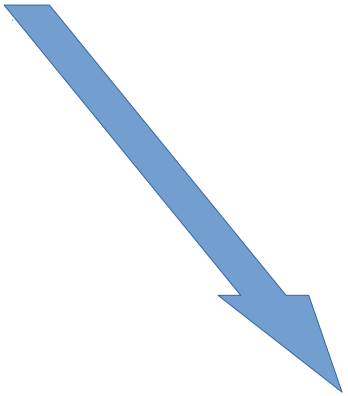
- Quais seriam os atributos que compõem a minha classe?
- Justamente as variáveis que foram apresentadas no slide anterior
- Vamos criar uma classe Java e adicionar os atributos.

Classe Random

```
public class Random {



}
```

Classe Random

```
public class Random {

}
```



Para que ele possa ser usada por outras classes.
Métodos e atributos também podem ser públicos.

Atributos da classe Random

```
public class Random {
    // esses são os parametros para geracao
    private long p = 2147483648L;
    private long m = 843314861;
    private long a = 453816693;

    private long xi = 1023; // essa eh a semente
}
```

Métodos da classe Random

```
public double getRand() {
    // calcula o proximo valor xi
    // calcula valor entre 0 e 1,
       dividindo por p
 }
```

Métodos da classe Random

```
public double getRand() {
      xi = (a + m * xi) % p;
    double d = xi; // promove p/ double
    return d / p;
}
```

Métodos da classe Random

```
public int getIntRand(int max)
{
    // gera valor entre [0 , 1)
    // multiplica por max
}
```

Métodos da classe Random

```
public int getIntRand(int max)
{
    double d = getRand() * max;
    return (int) d;
}
```

Usando a classe

- Vamos escrever um pequeno programa que gera números para a megasena
- Todo código está dentro de uma classe
- Antes de usar o gerador de números aleatórios é preciso “instanciar um objeto”

Classe MegaSena

```
public class MegaSena {
   public static void main(String[] args) {
      // cria um objeto para gerar números aleatórios
      for (int i = 0; i < 6; i++) {
            // Pega um número aleatório entre 1 e 60
            // mostra o número
      }
   }
}
```

Classe MegaSena

```
public class MegaSena {
  public static void main(String[] args) {
     // cria um objeto para gerar números aleatórios
     Random r = new Random();
       for (int i = 0; i < 6; i++) {
              // Pega um número aleatório entre 1 e 60
              // mostra o número
       }
  }
}
```

Classe MegaSena

```
public class MegaSena {
  public static void main(String[] args) {
     // cria um objeto para gerar números aleatórios
     Random r = new Random();
     for (int i = 0; i < 6; i++) {
            // Pega um número aleatório entre 1 e 60
           int k = r.getIntRand(60) + 1;
            // mostra o número
           System.out.println(i+1 + "o. Numero: " + k);=
     }
  }
}
```

Executando MegaSena

- 1o. Numero: 57
  2o. Numero: 29
  3o. Numero: 28
  4o. Numero: 14
  5o. Numero: 22
  6o. Numero: 11
- Qual o problema com esse programa?

Executando MegaSena

- 1o. Numero: 57
  2o. Numero: 29
  3o. Numero: 28
  4o. Numero: 14
  5o. Numero: 22
  6o. Numero: 11
- Qual o problema com esse programa?
  - Todas as vezes, a sequência é a mesma
- Como resolver isso?

ICMC USP SÃO CARLOS SSC

Solução 1

- Vamos permitir que a classe usuária modifique a semente a ser usada
- A semente pode, por exemplo, ser fornecida pelo próprio usuário
- Para isso vamos dar acesso ao atributo da semente
- public long xi = 1023;

Problema com solução 1

- O problema com essa solução é que ela dá acesso aos atributos internos da classe Random
- Uma das características de POO é ocultamento de informação
- Imagine, por exemplo que alguém precise mudar o nome da variável
- Uma classe deve ser acessada por meio da sua interface pública

Solução 2

- Vamos criar uma interface pública para alterar a semente
- Criamos um método setSemente(int k)
- Em vez de acessar diretamente a variável, invocamos esse método
- Toda vez que precisamos consultar ou alterar um atributo do objeto usamos **getters** e **setters**

Solução 2

```
private long xi = 1023;


public void setSemente(int semente) {
        xi = semente;
}
```

Solução 2 - MegaSena

```
public static void main(String[] args) throws Exception {

    Random r = new Random();

    System.out.print("Digite um número inteiro para
semente: ");

    int semente = EntradaTeclado.leInt();

    r.setSemente(semente);

    ……..
```

Construtor

- Quando criamos um objeto

```
r = new Random();
```

  é chamado um método especial da classe
- Esse método serve para inicializar os atributos do objeto
- Como podemos usar isso na nossa classe Random?

Construtor

- Quando criamos um objeto

```
r = new Random();
```

  é chamado um método especial da classe
- Esse método serve para inicializar os atributos do objeto
- Como podemos usar isso na nossa classe Random?
- Podemos passar parâmetros para o construtor

Construtor Random

```
public Random(int k)
{
    xi = k;
}
```

Construtor Random

```
public static void main(String[] args) throws
Exception {

    System.out.print("Digite um número inteiro para
                      semente: ");

    int semente = EntradaTeclado.leInt();

    Random r = new Random(semente);
```

Toda classe tem

- Toda classe tem um construtor
- Se você não declarar, é criado um, que não faz nada
- Se vc declarar um, o que não faz nada, deixa de existir
- Você pode ter mais do que um

Múltiplos construtores

```
public Random(int k)
{
    xi = k;
}

public Random() {
}
```

Múltiplos construtores

```
public Random(int k)
{
    xi = k;
}

public Random() {
}
```

```
Random r = new Random(2048);
```

```
Random r = new Random();
```

Exercícios

- Crie um programa que instancie dois objetos da classe Random, com duas sementes diferentes. Em seguida, vá gerando um número entre 0 e 500 com cada um deles, até que o número gerado seja o mesmo. Mostre qual é o número gerado e quantas iterações são necessárias.
- Modifique o construtor padrão da sua classe Random, de modo que se ele for usado para criar o objeto, então a semente vai ser inicializada com o valor aleatório dado por:

```
Calendar.getInstance().getTimeInMillis();
```

  Use essa nova versão em um dos objetos do programa anterior.

Exercício

- Implemente uma classe ObesidadePessoa com:
- atributos: peso e altura (ambos do tipo double)
- métodos:
  - setPeso: recebe peso por parâmetro e atualiza o atributo
  - setAltura: recebe altura por parâmetro e atualiza o atributo
  - getPeso: devolve peso
  - getAltura: devolve altura
  - calculaIMC: devolve Índice de Massa Corporal. IMC = peso/altura^2
  - defineObesidade = chama o método calculaIMC e exibe mensagem na tela, de acordo com a seguinte regra:
    - se IMC > 25, exibe “Risco de obesidade”, senão exibe “Não há risco de obesidade”