Departamento de Engenharias e Computação
Colegiado de Ciência da Computação
Disciplina: Linguagem de Programação III

Cap 2 – Conceitos Básicos de Orientação a Objetos

Parte 3 – Construtores

Professor: Otacílio José Pereira

Plano de Aula

- ***Objetivos***
  - Identificar o processo de criação de objetos
  - Compreender e aplicar construtores na implementação de classes

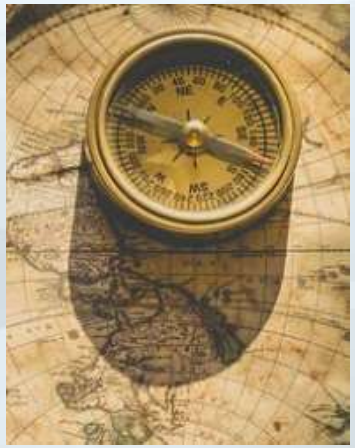
- ***Tópicos***
  - Contexto da disciplina
  - Refletindo sobre a criação de objetos
  - Declaração, instanciação e inicialização
  - Construtores
  - Construtores explícitos
  - Construtor implícito

Contexto

- Onde estamos?
- Foco agora!
- Cenário de exemplo

Onde estamos?

- *Considerando nosso planejamento inicial*



- Capítulo 1 – Introdução
- Capítulo 2 - Conceitos básicos de Orientação a Objetos
  - Parte 1 : Classes, objetos, atributos e métodos
  - Parte 2 : Encapsulamento, modif. de acesso, getters e setters
  - Parte 3 : Construtores
  - Parte 4 : Relação entre classes
- Capítulo 3 – Herança e Polimorfismo
- Capítulo 4 – Classes abstratas e interfaces
- Capítulo 5 – Generics, collections e outros tópicos
- Capítulo 6 – Desenvolvimento de um projeto em OO

Foco

Onde estamos?

- Neste momento que estamos discutindo os conceitos básicos de OO, organizamos nossa trajetória nos seguintes tópicos
  - **(Ok) Parte 1**: Classes, objetos, atributos e métodos e discussões iniciais sobre orientação a objetos
  - **(Ok) Parte 2**: Encapsulamento, modificadores de acesso, getters e setters e sobrecarga de métodos
  - **(Foco agora) Parte 3**: Construtores e instanciação de objetos
  - **Parte 4**: Atributos e métodos de classe
  - **Parte 5**: Outros elementos básicos

Refletindo

- Situação para análise
- Declaração de variáveis
- Instanciação de objetos
- Inicialização dos valores

Observe o código a seguir

- Considerando a implementação da classe Conta à esquerda, analise as duas versões de código main
  - Em que momento que o objeto está sendo criado?
  - Quais os valores iniciais do objeto referenciado por c1?
  - Em termos práticos há alguma diferença entre eles?
    - Na criação dos objetos? - Na inicialização de valores?

```
class Conta {
    int Numero    = 1;
    int Agencia   = 1;
    String Titular;
    double Saldo  = 0.0;

    public int getAgencia() {
        return Agencia;
    }
```

```
public static void main(String[] args)
{
    Conta c1;
    c1 = new Conta();
    c1.setNumero(10);
    c1.setAgencia(30);
    c1.setTitular("Carla");
    c1.setSaldo(0.0);

    c1.Depositar(1000.0);
    c1.Sacar(300);
    c1.Imprimir();
}
```

```
public static void main(String[] args)
{
    Conta c1 = new Conta();
    c1.setTitular("Carla");
    c1.setSaldo(0.0);

    c1.Depositar(1000.0);
    c1.Sacar(300);
    c1.Imprimir();
}
```

Outras reflexões

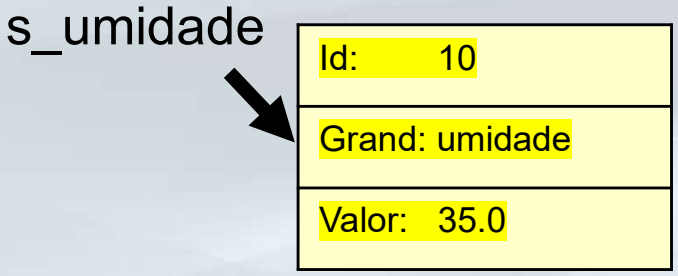
- Em nossas codificações anteriores, quando usamos o "comando" new?
  - Qual diferença da chamada no new em relação ao caso anterior? (Sem contar que tratam de classes e objetos diferentes)

```
public static void main (String args[])
{
    int     idSensor;

    Scanner scan = new Scanner(System.in);

    System.out.println(" Digite o Id do Sensor : ");
    idSensor = scan.nextInt();
```

- O que aconteceria neste caso?

Conta c1;

c1.setNumero(10);



Declaração, Instanciação e Inicialização

- Observando um exemplo
- Declaração
- Instanciação
- Inicialização
- Exercício e curiosidade

Declaração, instanciação e inicialização

- Dentro do que vimos, os objetos em uma linguagem Orientada a Objetos são acessados através de variáveis

- Declaração da variável
  - Uma variável, como sempre vimos em programação, é declarada com base em um tipo, ou no caso classe
  - Exemplo: Conta c1; Sensor s_umidade;

  - **Mas isso não quer dizer que já temos o objeto, temos apenas uma referência para um espaço que deve ser o objeto daquela classe**

```
Conta c1;
c1 = new Conta();
c1.setNumero(10);
c1.setAgencia(30);
c1.setTitular("Carla");
c1.setSaldo(0.0);
```

Declaração, instanciação e inicialização

- **Instanciação**
  - Para criar um objeto, deve ser realizada uma instanciação
  - No caso utiliza-se o “comando” **new**
  - Exemplo: c1 = new Conta();
  - Agora sim, há um objeto instanciado que pode ter seus atributos modificados
- **Declaração e instanciação juntos**
  - É comum realizarmos a declaração de uma variável e instanciação de um objeto juntos
  - Exemplo: Conta c1 = new Conta();

```
public static void main(String[] args)
{
    Conta c1;
    c1 = new Conta();
    c1.setNumero(10);
    c1.setAgencia(30);
    c1.setTitular("Carla");
    c1.setSaldo(0.0);

    c1.Depositar(1000.0);
    c1.Sacar(300);
    c1.Imprimir();
}
```

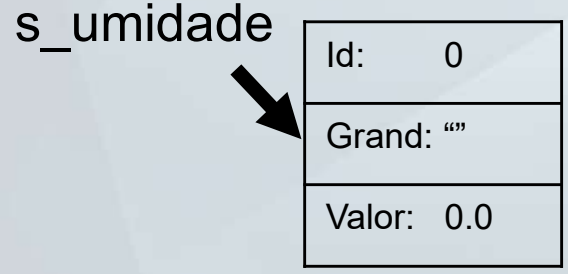
```
public static void main(String[] args)
{
    Conta c1 = new Conta();
    c1.setTitular("Carla");
    c1.setSaldo(0.0);

    c1.Depositar(1000.0);
    c1.Sacar(300);
    c1.Imprimir();
}
```

Declaração, instanciação e inicialização

- **Instanciação**
  - No momento da instanciação é que de fato surge o objeto em memória
  - Ou costumamos dizer que surgiu a instância da classe

- Ok, resolvemos a questão de criarmos o objeto, mas qual valor os atributos terão?

```
s_umidade = new Sensor();
```




```
public static void main(String[] args)
{
    Conta c1;
    c1 = new Conta();
    c1.setNumero(10);
    c1.setAgencia(30);
    c1.setTitular("Carla");
    c1.setSaldo(0.0);

    c1.Depositar(1000.0);
    c1.Sacar(300);
    c1.Imprimir();
}
```

```
public static void main(String[] args)
{
    Conta c1 = new Conta();
    c1.setTitular("Carla");
    c1.setSaldo(0.0);

    c1.Depositar(1000.0);
    c1.Sacar(300);
    c1.Imprimir();
}
```

Declaração, instanciação e inicialização

- **Inicialização**
  - Quais são os valores iniciais do objeto conta instanciado?
    - São os valores padrões do tipo: String vazia para titular e 0 (zero) para saldo
  - Estes valores podem ser alterados
  - Exemplo: c1.setNome(“Teobaldo Bisâncio”);
    c1.setSaldo(0);

- Declaração, instanciação e inicialização juntos
  - É possível em alguns casos realizar tudo em uma linha só
  - Exemplo: Conta c1 = new Conta(“Teobaldo Bisâncio”, 0.0);

  - Scanner s = new Scanner(System.in);

Visualizando graficamente

(Declaração) **(Instanciação)** **(Inicialização)**

```
Sensor s_umidade;
```

s_umidade

```
s_umidade = new Sensor();
```

s_umidade

Id:	0
Grand:	“”
Valor:	0.0

```
s_umidade.id = sc.nextInt();
```

s_umidade

Id:	10
Grand:	umidade
Valor:	35.0

Exercício e curiosidade

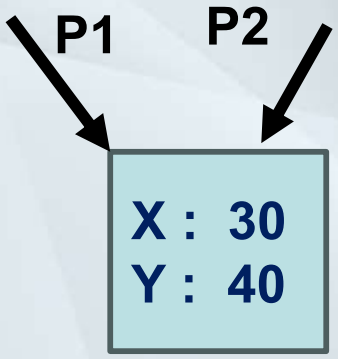
- Qual valor será impresso no programa a seguir?

```
class Ponto { int x, y; }
Ponto p1, p2;

p1 = new Ponto();     p1.x = 10;          p1.y = 20;
p2 = p1;
p2.x = 30;                p2.y = 40;

System.out.printf(“ P1 :   %d   %d  “, p1.x, p1.y);
System.out.printf(“ P2 :  %d   %d   “, p2.x, p2.y);
```

Exercício e curiosidade

- Perceba que a variável funciona de forma análoga a um ponteiro em C.

```
class Ponto { int x, y; }
Ponto p1, p2;
p1 = new Ponto();        p1.x = 10;          p1.y = 20;
p2 = p1;
p2.x = 30;               p2.y = 40;
System.out.printf(“ P1 :   %d   %d  “, p1.x, p1.y);
System.out.printf(“ P2 :  %d   %d   “, p2.x, p2.y);

TPONTO    *p1, *p2;
p1 = malloc(sizeof(TPONTO));
p2 = p1;         p2.x = 30;
```



Tipos por valor e por referência

- Exemplo com inteiros
- Tipos por valor
- Tipos por referência

Resumindo e refletindo

- Bom, vimos que nossas variáveis precisam ser declaradas e os objetos criados

- Para isso usamos o comando new

- Mas observe e analise os exemplos a seguir

```
int          a = 10;          b = a;    b = 20;
double       valor = 2.50;
```

- Por que eles não usam o comando new?
- Como a questão dos objetos são tratados?
- E quais os valores de a e b no exemplo acima?

Tipos por valor e por referência

- O que acontece quando você realiza as seguintes declarações?
  - **Caso A (tipos simples):**
  - int i;

  - **Caso B (estruturas em C):**
  - struct Ponto { float x, y; }
  - struct Ponto p1;

  - **Caso C (Classes em Java):**
  - Class Ponto { float x, y; }
  - Ponto p1;

- Casos A e B: aplicam o conceito de tipo/variável por valor
  - Ocorre a declaração, instanciação e inicialização padrão
- Casos C: aplicam o conceito de tipo/variável por referência
  - Ocorreu no caso apenas a declaração. Não há espaço alocado em memória.

Tipos por valor e por referência

- Considerando então as diferenças expostas, o que aconteceria com os exemplos abaixo?

  – **Caso B (Estruturas em C) :**

```
struct Ponto { float x, y; }
struct Ponto p1;
...
p1.x = 10;    p1.y = 20;
```

  – **Caso C (Classes em Java) :**

```
Class Ponto { float x, y; }
Ponto p1;
p1.x = 10;     p1.y = 20;
```

Tipos por valor e por referência

- Tipos por valor em Java
  - **Alguns exemplos de tipos por valor em Java são**

byte
short
char
float
double
int
long
boolean

```
public static void main(String[] args) {
      byte tipoByte = 127;
      short tipoShort = 32767;
      char tipoChar = 'C';
      float tipoFloat = 2.6f;
      double tipoDouble = 3.59;
      int tipoInt = 2147483647;
      long tipoLong = 92233720368547758O7L;
      boolean tipoBooleano = true;
```

Tipos por valor e por referência

- Tipos por referência
  - São os tipos baseados na criação de classes
- Perceba o caso da classe Ponto do exemplo dado
- No caso é necessário criar a construção explícita do objeto via comando **new Ponto()**
  - **Caso B:**

```
struct Ponto { float x, y; }
struct Ponto p1;
...
p1.x = 10;    p1.y = 20;
```

  - **Caso C:**

```
Class Ponto { float x, y; }
Ponto p1 = new Ponto();
p1.x = 10;     p1.y = 20;
```

Construtores

- Introdução
- Construtor Padrão
- Outros construtores
- Construtor implícito

```
public class Conta {
    int Agencia;
    int Conta;
    String NomeTitular;
    float Saldo;

    Conta()
    {
        Agencia     = 0;
        Conta       = 0;
        NomeTitular = "Novo cliente";
        Saldo       = 0;
    }
}
```

```
public static void main(String[] args) {
    Conta c;

    c = new Conta();

    c.Conta = 7632;
    c.Agencia = 232;
    c.NomeTitular = "Teobaldo Bisâncio";
}
```

Construtores

- Todo objeto para ser instanciado e inicializado necessita de um método especial da classe denominada “construtor”
- No exemplo visto, vimos que tínhamos duas formas de instanciar e inicializar o objeto conta
- Sem informar valores, deixando que valores implícitos (em geral valores padrões dos tipos) sejam empregados
  - Conta c1 = new Conta();
- Informando os valores a serem empregados na inicialização do objeto
  - Conta c1 = new Conta(“Teobaldo Bisâncio”, 0);

Construtor Padrão

- Um construtor padrão não apresenta parâmetros, ele utiliza valores indicados no método construtor ou, caso estes não sejam informados, são empregados os valores padrões dos tipos
  - Por exemplo, para inteiro o valor padrão é 0 (zero)

- Exemplo

```
public class Conta {
    int Agencia;
    int Conta;
    String NomeTitular;
    float Saldo;

    Conta()
    {
        Agencia     = 0;
        Conta       = 0;
        NomeTitular = "Novo cliente";
        Saldo       = 0;
    }
}
```

```
public static void main(String[] args) {
    Conta c;

    c = new Conta();

    c.Conta = 7632;
    c.Agencia = 232;
    c.NomeTitular = "Teobaldo Bisâncio";
}
```

- No exemplo acima
  - Quais os valores dos atributos Nome e Saldo de um novo objeto?

Outros construtores

- Juntamente com o construtor padrão, um objeto pode ser instanciado e inicializado por meio de outros construtores que apresentam seus parâmetros estabelecidos
- Exemplo

```
public static void main(String[] args) {
    Conta c;

    c = new Conta(7632, 232, "Teobaldo Bisâncio");

}
```

```
public class Conta {

    int Agencia;
    int Conta;
    String NomeTitular;
    float Saldo;

    Conta()
    {
        Agencia     = 0;
        Conta       = 0;
        NomeTitular = "Novo cliente";
        Saldo       = 0;
    }

    Conta(int pConta, int pAgencia, String pNome)
    {
        Agencia     = pAgencia;
        Conta       = pConta;
        NomeTitular = pNome;
        Saldo       = 0;
    }

}
```

- No exemplo acima
  - Quais os valores dos atributos Nome e Saldo de um novo objeto?
  - O que difere este construtor do construtor padrão?

Construtor implícito

- Os construtores anteriores são ditos explícitos, pois nós criamos explicitamente me nossas classes.
- Quando não falamos nada a respeito da construção de um objeto, o Java cria automaticamente um construtor que cria o objeto
- Este construtor criado automaticamente é chamado de construtor implícito
- Ele não aparece no código, ele é criado implicitamente para que os objetos sejam criados

Cadê o construtor?

```
12  class Conta {
13      int Numero;
14      int Agencia;
15      String Titular;
        double Saldo;

        public void Sacar(double pValor)
19      {
20          Saldo -= pValor;
21      }
```

Exercício

- Exercício 7
  - Discuta quais potenciais construtores para a classe **Conta**

- Exercício 8
  - Acrescente o construtor padrão para a classe **Conta**
  - Acrescente o construtor com parâmetros para a classe **Conta**
  - Em cada construtor, coloque uma mensagem que sinalize que aquele construtor foi ativado.
    - Exemplo: System.out.print(“ Construtor padrão ativado”);

Construtores

- Pontos chaves do exercício
  - Classe Conta

```
public class Conta {

    private int     Agencia;
    private int     NumeroConta;
    private String  NomeTitular;
    private double  Saldo = 0;

    Conta() // Construtor padrão
    {
        Agencia         = 0;
        NumeroConta     = 0;
        NomeTitular     = "Novo titula";
        Saldo           = 0;
    }

    Conta(int    pAgencia,     int    pNumeroConta,
          String pNomeTitular, double pSaldo)

    {
        Agencia         = pAgencia;
        NumeroConta     = pNumeroConta;
        NomeTitular     = pNomeTitular;
        this.Depositar(pSaldo);
    }
```

Classe Banco

- Uso dos construtores

```
Conta c2 = new Conta(1806, 734210, "José Teolbaldo", 150);
c2.Imprimir();
```

Exercício

- Se der tempo, faça o exercício a seguir
  - Crie uma classe ponto
  - Ela possui os atributos X e Y
  - Crie os construtores padrão e um com parâmetros
  - Crie os métodos getters e setters
  - Crie os métodos
    - Deslocar : Que muda o ponto de um deslocamento dX e dY
    - Distância : que calcula a distância do ponto a um outro ponto de referência
  - Realize alguns testes com variáveis do tipo ponto

Conclusões

- Retomando Plano de Aula
- Revisão
- Para saber mais

Conquistamos a aula?

- ***Objetivos***
  - Identificar o processo de criação de objetos
  - Compreender e aplicar construtores na implementação de classes

- ***Tópicos***
  - Contexto da disciplina
  - Refletindo sobre a criação de objetos
  - Declaração, instanciação e inicialização
  - Construtores
  - Construtores explícitos
  - Construtor implícito

Departamento de Engenharias e Computação
Colegiado de Ciência da Computação
Disciplina: Linguagem de Programação III

Cap 2 – Conceitos Básicos de Orientação a Objetos

Parte 3 – Construtores

Professor: Otacílio José Pereira