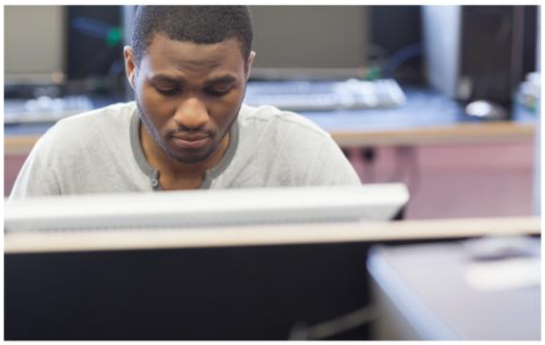

ORACLE
Academy

Java Fundamentals

4-2

Classes de Objeto e Piloto



ORACLE
Academy

Objetivos

- Esta aula abrange os seguintes tópicos:
  - Descrever o formato geral de um programa Java
  - Descrever a diferença entre uma classe de Objeto e uma classe de Driver
  - Acessar no mínimo duas APIs de classe Java
  - Explicar e dar exemplos de palavras-chave Java
  - Criar uma classe de Objeto
  - Criar uma classe de Driver

ORACLE Academy

Formato Geral do Programa Java

- Há dois formatos gerais de classes Java:
  - Classes de driver
  - Classes de objeto

Classes de Driver

- Classes de driver:
  - Contêm um método main
    - É necessário um método main para executar um programa Java
    - O método main pode incluir:
      - Instâncias de objetos de uma classe de objeto
      - Variáveis
      - Loops, instruções condicionais (if-else)
      - Outra lógica de programação
  - Também pode conter outros métodos estáticos

Se você já tiverem abordado o Greenfoot, main() é semelhante a act() no sentido de que é onde o programa se inicia. No entanto, diferente do Greenfoot, o programa Java executará apenas um método main() enquanto um programa Greenfoot pode ter muitos métodos act() ativos.
Um projeto Java pode ter mais de uma classe de driver, isto é, uma para cada classe de objeto para testar e depurar, e uma para iniciar o programa principal.

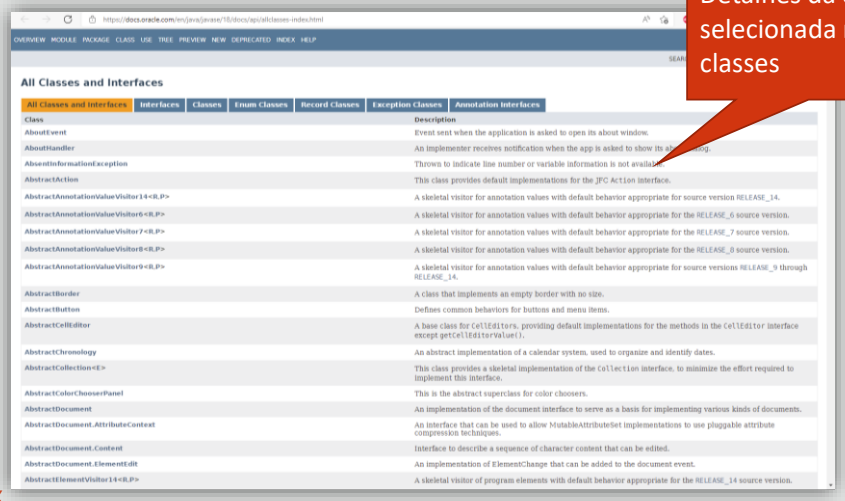
Classes de Objeto

- Classes de objeto:
  - São classes que definem objetos a serem usados em uma classe de driver
  - Podem ser encontradas na API do Java ou criadas por você
  - Exemplos: String, BankAccount, Student, Rectangle

A API do Java é uma biblioteca de pacotes e classes de objeto que já estão escritas e estão disponíveis para uso em seus programas.

The Java API

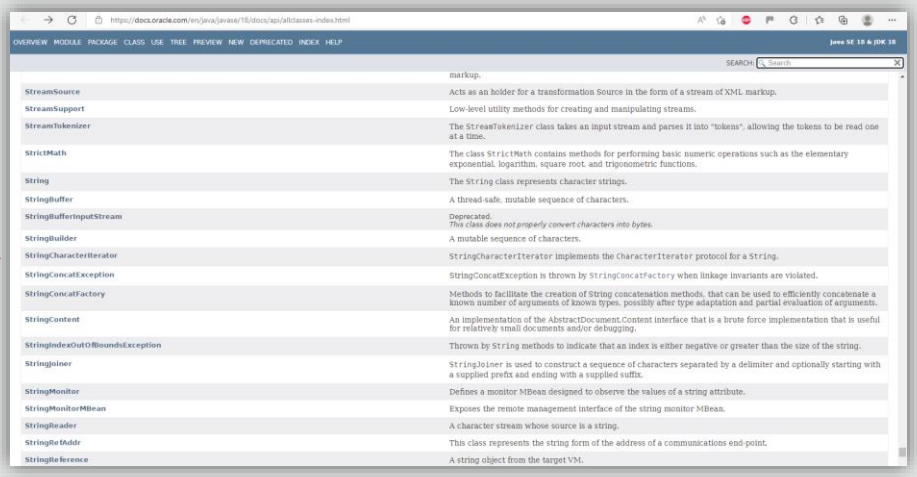
- A API do Java
  - https://docs.oracle.com/en/java/javase/18/docs/api/allclasses-index.html



A API do Java: Classe String

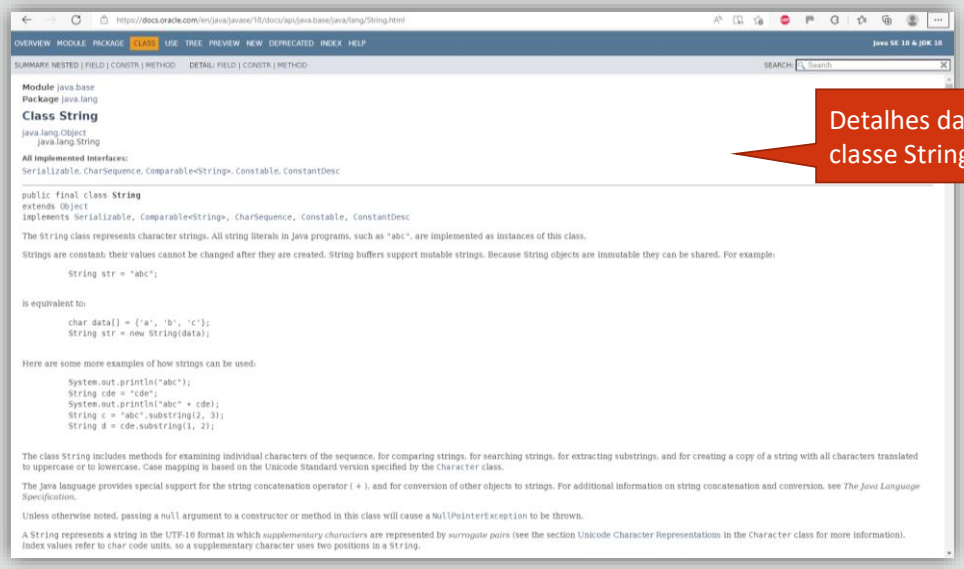
- Percorra a lista de classes até ver String, em seguida, clique no link
- Você verá o seguinte:

Role para baixo nesta lista até String



A API do Java: Examinar a Classe String

- Examine e discuta os construtores e métodos



A API do Java: A Classe String

- Usaremos a classe String em programas
- O construtor mais comum para esta classe é:
  - String(String original)
- Os Métodos Comuns incluem:

Método	Descrição
charAt(int index)	Retorna o valor de char no índice especificado.
length()	Retorna o comprimento dessa string.
substring(int beginIndex)	Retorna uma nova string que é uma substring desta string.

Uma Classe de Objeto Simples Criada pelo Programador: Student

- A classe Java é usada para armazenar ou representar dados do objeto que a classe representa
- Existem várias classes já disponíveis da API do Java, mas você poderá criar mais
- Por exemplo, podemos criar um modelo, ou representação programática, de um Aluno
- As informações que poderemos precisar para um aluno incluem Id do Aluno, Nome e Média Geral
- Nesta aula, criaremos uma classe de objeto chamada Student, em seguida, uma classe de driver chamada StudentTester

Uma Classe de Objeto Simples Criada pelo Programador: Student

- Todas as classes Java são criadas em um arquivo de texto com o mesmo nome que a classe
- Esses arquivos também têm uma extensão .java
- Nesta aula, criaremos uma classe Student e uma classe StudentTester no um IDE Java

O Eclipse criará e nomeará automaticamente os arquivos de código-fonte .java e, após a compilação, os arquivos de código de byte .class.

Sintaxe para Criar uma Classe de Objeto Simples Criada pelo Programador

- Veja a seguir um exemplo de sintaxe para criar uma classe de objeto criada pelo programador
- As palavras-chave Java são:
  - package (opcional)
  - import (opcional)
  - public class

```
package <package_name>;
import <other_packages>;
public class ClassName
{
   <variables (also known as fields)>;
   <constructor method(s)>;
   <other methods>;
}
```

ORACLE Academy

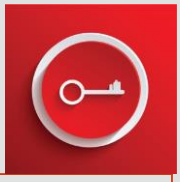
As variáveis de classe, às vezes, são denominadas de "variáveis de instância" para distingui-las das "variáveis de local".

Termos-chave

Termo	Definição
palavra-chave package	• Define a localização desta classe em relação a outras classes, e fornece um nível de controle de acesso. • Use modificadores de acesso (como public e private) para controlar o acesso.
palavra-chave import	• Define outras classes ou grupos que você está usando em sua classe. • A instrução de importação fornece as informações do compilador que identificam as classes externas usadas na classe atual.
palavra-chave class	• Precede o nome da classe. • O nome da classe e o nome do arquivo devem coincidir quando a classe for declarada como pública (o que é uma boa prática). • No entanto, a palavra-chave public na frente da palavra-chave class é um modificador e não é necessária.

Os pacotes são abordados detalhadamente em Programação Java.

Termos-chave



Termo	Definição
variáveis de classe ou campos de instância (muitas vezes abreviado para campos)	• Variáveis ou os dados associados a programas (como integers, strings, arrays e referências a outros objetos).
Construtores	• Métodos chamados durante a criação (instanciação) de um objeto (uma representação na memória de uma classe Java.)
Métodos	• Os métodos podem ser executados em um objeto • Eles também são chamados de métodos de instância • Os métodos podem retornar valores da variável de objeto (às vezes chamada de funções) ou podem alterar os valores da variável de um objeto.

Palavras-chave Java

- Todos os programas Java usam palavras-chave Java
- Os exemplos incluem as seguintes palavras: class, public, String, int, for, while e double
- A cor da fonte das palavras-chave mudará no Java IDE
- Lista de palavras-chave Java:
  - https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

A palavra-chave Java é uma palavra que tem uma função especial na linguagem Java e não pode ser usada como nomes de classes, métodos ou variáveis.

Outro nome para palavra-chave é "palavra reservada".

Convenções de Nomenclatura do Programa Java

- Convenções de Nomenclatura para um programa Java:
  - O nome do pacote opcional é definido antes de uma instrução de importação na concatenação com minúsculas
  - As instruções de importação opcionais são definidas abaixo do nome do pacote
  - O nome da classe é um nome com label que usa concatenação com maiúscula

Concatenação

- A concatenação é a prática de criar strings com palavras juntas em maiúsculas e sem espaço
- A concatenação com minúsculas gera strings de palavras juntas em maiúsculas, mas a palavra principal não fica em maiúscula
- Por exemplo:
  - thisIsLowerCamelCase
- A concatenação em maiúscula gera strings de palavras juntas em maiúsculas com a palavra principal em maiúscula
- Por exemplo:
  - ThisIsUpperCamelCase

Convenções de Nomenclatura Adicionais para um Programa Java

- Convenções de nomenclatura Adicionais para um programa Java:
  - Os nomes de variáveis são curtos, mas significativos em concatenação com minúsculas
  - Os nomes de constantes são declarados em letras maiúsculas com o modificador final
  - Os construtores têm o mesmo nome que a classe
  - Métodos são verbos nomeados em concatenação com minúsculas

Exemplos de Convenções de Nomenclatura

- As diretrizes para uma classe de objeto criada pelo programador são identificadas neste exemplo de Student
  - Crie este arquivo no um IDE Java //é o símbolo para os comentários

```
package com.example.domain;  // Package Declaration
import java.util.Scanner;    // An Import Statement for other packages
public class Student {       // Class Declaration for this file
   private int studentId;    // Variable Declarations for this class
   private String name;
   private String ssn;
   private double gpa;
   public final int SCHCODE = 34958// A Constant Declaration
   public Student(){          // A Construtor
   }//fim construtor
   public int getStudentId(){ // An accessor Method
      return studentId;
   }//fim do método studentId
   public void setStudentId(int x){ // A mutator Method
      studentId = x;
   }//fim do método setStudentId
}//fim da classe Student
```


Muitos métodos têm nomes que se iniciam com set, get, is, has, etc.

Uma Classe de Objeto Simples Criada pelo Programador: Student

- A palavra-chave opcional do pacote é usada em Java para agrupar classes
  - Um pacote é implementado como um filtro
  - Como uma pasta, ele fornece um namespace a uma classe
  - Recomenda-se sempre declarar um pacote na parte superior da classe

```
package com.example.domain;
```

Uma Classe de Objeto Simples Criada pelo Programador: Student

- No exemplo a seguir, uma classe Student e uma classe Teacher poderiam estar em uma pasta sob o nome de domínio de cada desenvolvedor
- Se uma empresa chamada Acme tiver desenvolvedores chamados Smith e Jones, os pacotes poderão ser:
  - pacote com.acme.smith
  - pacote com.acme.jones
- O caminho para o arquivo de Jones é mostrado a seguir

View do Namespace para Jones:
- com.acme.jones
  - Student
  - Teacher

View da Pasta para Jones:
```
+com
|_+acme
  |_+jones
    |_+Student.java
    |_+Teacher.java
```

Palavra-chave import

- A palavra-chave import é usada para identificar pacotes ou classes de objeto que você deseja usar em sua classe
- Você pode importar uma única classe ou um pacote inteiro
- Você pode incluir várias instruções de importação
- As instruções de importação seguem a instrução do pacote e precedem a declaração da classe

```
import java.util.Scanner;
```

Instruções de Importação

- Uma instrução de importação não é obrigatória e, por padrão, sua classe sempre importará java.lang da API
  - https://docs.oracle.com/en/java/javase/18/docs/api/java.base/java/lang/Package.html
- Você não precisa importar as classes que estão no mesmo pacote que a instrução de importação

Exemplos de Instrução de Importação

- Exemplos adicionais de instruções de importação:

```
import java.util.Date;  //Import the individual class.
import java.util.*;     //Import the entire package.
import java.util.Date;  //Import using multiple statements.
import java.util.Calendar;
```


Embora * importe todas as classes de um pacote, ele poderá ser útil nomeiem as classes específicas no pacote e aprendam com mais facilidade.

Variáveis da Classe Student

- Além da instrução do pacote e das instruções de importação, a classe Student conterá variáveis de Id do aluno, nome, cpf, médias das notas e código da escola
- Isso exigirá a definição de uma classe com variáveis de classe e um construtor
- Além disso, serão adicionados métodos que podem acessar e alterar as variáveis

Não confunda o uso de "variável de classe" do slide com campos com variáveis estáticas, que também são chamados de variáveis de classe.

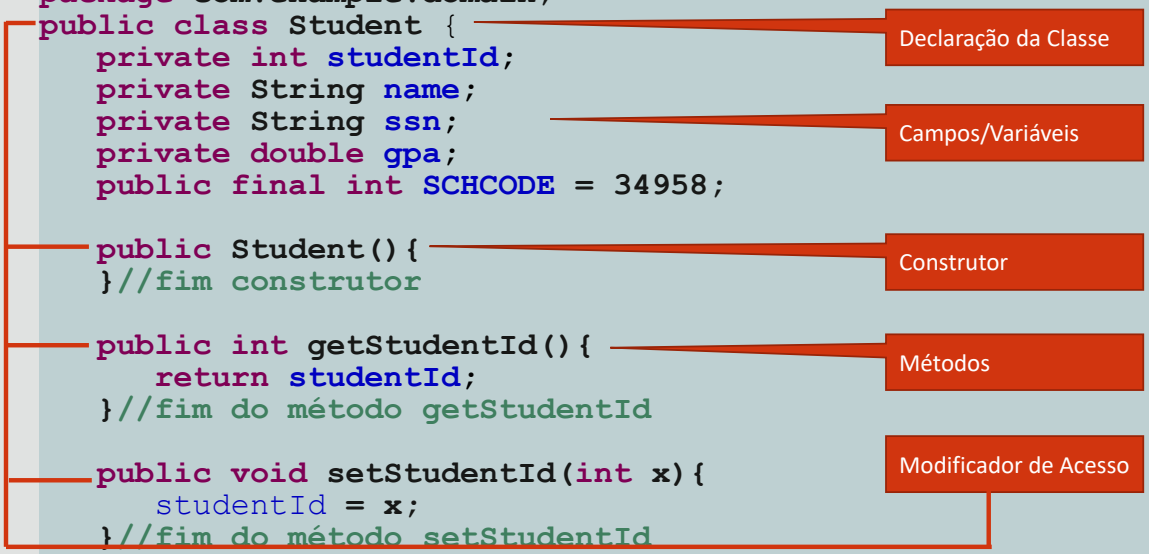
A Classe Student

```
package com.example.domain;
public class Student {
   private int studentId;
   private String name;
   private String ssn;
   private double gpa;
   public final int SCHCODE = 34958;

   public Student(){
   }//fim construtor

   public int getStudentId(){
      return studentId;
   }//fim do método getStudentId

   public void setStudentId(int x){
      studentId = x;
   }//fim do método setStudentId
}//fim da classe Student
```



Convenções da Declaração da Classe

- Convenções da declaração da classe:
  - O nome deve começar com um caractere e deve conter números, _ ou $
  - Use a concatenação com maiúsculas se o nome tiver mais de uma palavra
  - Para uma classe de objeto simples criada pelo programador, o modificador de acesso deve ser public (todos os modificadores de acesso são public, private ou protected)

```
public class Student{}
```

Todo o código desta classe deve ser colocado entre chaves { }.

ORACLE Academy

Os identificadores nomeados pelo usuário raramente usam $ ou _ em Java.

Convenções das Variáveis da Classe

- Convenções das variáveis da classe:
  - Variáveis de classe devem ser declaradas com o modificador de acesso privado para proteger os dados
  - As variáveis de classe são nomeadas usando a concatenação com minúsculas
  - Uma exceção é uma constante (um valor que não muda) que deve ser nomeada usando letras maiúsculas e declarada como pública para permitir que programas de driver acessem o valor

Proteger os valores dos campos com privado é conhecido como encapsulamento. Esse é um dos princípios básicos de uma linguagem de programação orientada a objetos, juntamente com herança e polimorfismo.

Exemplos de Declaração de Variáveis de Classe

- Exemplos de declaração de variáveis de classe:
  - `private int length;`
  - `private int width;`
  - `private double area;  //constant`
  - `public final double SCALE = 0.25;`
  - `private String name;`

Métodos do Construtor

- Um métodos do construtor é exclusivo em Java, pois:
  - O método cria uma instância da classe
  - Construtores sempre têm o mesmo nome que a classe e não declarar um tipo de retorno

```
public Student()
{

}//fim construtor
```

Todo o código deste método deve ser colocado entre chaves { }.

Colchetes também podem ser chamados de chaves.

Métodos do Construtor

- Com métodos do construtor:
  - Você pode declarar mais de um construtor em uma declaração de classe
  - Não é necessário declarar um construtor, o Java fornecerá um construtor (em branco) padrão para você
  - Se você declarar um ou mais construtores, o Java não fornecerá um construtor padrão

```
public Student()
{

}//fim construtor
```

Todo o código deste método deve ser colocado entre chaves { }.

Construtores sem Parâmetros

- Se você criar um construtor sem parâmetros (o parêntese fica vazia), você pode deixar o conteúdo do construtor (entre o { e }) em branco
- Este é chamado de construtor padrão e será o mesmo construtor fornecido pelo Java se você não declarar um
- Esse construtor inicializa as variáveis de classe numéricas como zero as e variáveis de objeto (como Strings) como null

```
public Student()
{

}//fim construtor
```

Sem parâmetros

Construtores sem Parâmetros

- Se você criar um construtor sem parâmetros (o parêntese fica vazia), você também poderá inicializar as variáveis entre o { e }
  - Este também é chamado de construtor padrão
  - Esse construtor inicializa as variáveis de classe numéricas como zero as e variáveis de objeto (como Strings) como null
  - Ele tem os mesmos resultados que o slide anterior, mas os valores são mais evidentes

```
public Student(){
    studentId = 0;
    name = "";
    ssn = "";
    gpa = 0.0;
}//fim construtor
```

Sem parâmetros

Construtores com Parâmetros

- Se você criar um construtor com parâmetros (o parêntese NÃO ficará vazia), você também poderá inicializar as variáveis entre o { e }
- Esse construtor inicializará as variáveis de classe com os valores que são enviados a partir da principal classe do driver

```
public Student(int x, String n, String s, double g){
   studentId = x;
   name = n;
   ssn = s;
   gpa = g;
}//fim construtor
```

Exemplo de Construtor Padrão

- O método do construtor, neste exemplo, é um construtor padrão que cria uma instância de Student

```
package com.example.domain;
public class Student
{
   private int studentId;
   private String name;
   private String ssn;
   private double gpa;
   public final int SCHCODE = 34958;

   public Student(){
   }//fim construtor
   public int getStudentId(){
      return studentId;
   }//fim do método getStudentId
}//fim da classe Student
```

Construtor

Exemplo de Construtor Padrão

- O exemplo de Student no slide anterior ilustra um simples construtor sem argumentos
- O valor retornado do construtor é uma referência a um objeto Java do tipo criado
- Lembre-se, os construtores também podem usar parâmetros

Construtores

- Um construtor é um método que cria um objeto
- Em Java, os construtores são métodos com o mesmo nome que suas classes usados para criar uma instância de um objeto
- Construtores são invocados usando a nova palavra-chave
- Exemplo de código que poderia ser usado em uma Classe de Driver para criar um objeto do construtor Student:

```
Student stu = new Student();
```


Os construtores não têm um tipo de retorno porque eles sempre retornam uma nova instância da classe (um objeto).

Método Main

- Para executar um programa Java, é necessário definir o método main de uma classe de driver
- O método main é automaticamente chamado quando a classe é chamada
- Lembre-se de dar ao arquivo o mesmo nome da classe

Você pode fazer com que o Java IDE crie o método main() para você quando a classe for criada.

Uma Classe de Driver Simples Criada Pelo Programador: StudentTester

- Nos exemplos deste curso, geralmente usaremos o nome da classe de objeto seguido da palavra "Tester"
- Veja a seguir, um exemplo de uma classe de driver Java simples chamada StudentTester com um método main

```
public class StudentTester
{
   public static void main(String args[])
   {
   }//fim do método main
}//fim da classe StudentTester
```

Uma Classe de Driver Simples Criada Pelo Programador: StudentTester Exemplo 1

- Neste exemplo, uma instrução será adicionada para criar um objeto de Aluno
- Um aluno é criado e as variáveis de classe são inicializadas conforme descrito anteriormente sob construtores padrão

```
public class StudentTester
{
   public static void main(String args[])
   {
      Student s1 = new Student();
   }//fim do método main
}//fim da classe StudentTester
```

Uma Classe de Driver Simples Criada Pelo Programador: StudentTester Exemplo 2

- Neste exemplo, a instrução para criar um objeto da classe Student é diferente
- Este Aluno é criado usando parâmetros
- Você consegue adivinhar a Id deste aluno? Nome? CPF? Média Geral?
- Adicione outro aluno usando o construtor padrão

```
public class StudentTester
{
   public static void main(String args[]
   {
  Student s1 = new Student(123, "Mary Smith", "999-99-9999", 3.4);
   }//fim do método main
}//fim da classe StudentTester
```

Aprimorando a Classe de Objeto Student

- A maioria dos ambientes de desenvolvimento integrado modernos oferecem uma maneira fácil de gerar automaticamente os métodos do acessador (getter) e do modificador (setter)
- Outro método útil durante os testes, a criação e a modificação de objetos é o método toString ()
- Nos próximos slides, desenvolveremos novos métodos para a classe de objeto Student e modificaremos a classe de driver StudentTester para testá-las

Métodos de Acessador e Modificador

- É comum criar um conjunto de métodos que manipulam e recuperam valores de dados de classe:
  - Acessadores (getters):
    - Os métodos que retornam (get) o valor de cada variável de classe
  - Modificadores (setters):
    - Os métodos que alteram (get) o valor de cada variável de classe

Eles são necessários, pois, em geral, temos campos privados. Além de simplesmente definirem valores de campos, os modificadores (setters) também podem ser usados para validar dados.

Métodos de Acessador e Modificador para Aprimorar a Classe de Objeto Student

- Exemplos (a seguir):
  - Adicione os seguintes métodos de Acessador:
    - getStudentId()
    - getName()
    - getSSN()
    - getGPA()
  - Adicione os seguintes métodos de Modificador:
    - setStudentId()
    - setName()
    - setSSN()
    - setGPA()
  - Adicione um método toString() à classe Student que nos permitirá ver os dados do Aluno como resultado

Aprimorando a Classe de Objeto Student

- Adicione o método getStudentId() e o método setStudentId() conforme mostrado a seguir
- O nome do objeto da classe (this) é usado para distinguir entre a variável StudentId da classe e o parâmetro StudentId sendo passado como argumento

```
public int getStudentId(){
   return studentId;
}//fim do método getStudentId

public void setStudentId(int studentId){
   this.studentId = studentId;
}//fim do método setStudentId
```


O "this" é necessário quando o parâmetro tem o mesmo nome que o campo. "this" identifica o campo.

Métodos Adicionais para a Classe de Objeto Student

- Agora, usando o exemplo do slide anterior, adicione os seguintes métodos à classe de objeto Student:
  - getName()
  - getSSN()
  - getGPA()
  - setName()
  - setSSN()
  - setGPA()

Adicionar o Método toString() à Classe de Objeto Student

- Agora, adicione o método toString() à classe de objeto Student
- Observe que qualquer objeto de String pode ser criado por você para exibir as informações sobre cada aluno:

```
public String toString()
{
   String s1 = "";
   s1 =  "Student Id: " + getStudentId()
       + "Student Name: " + getName()
       + "Student SSN: " + getSSN()
       + "Student GPA: " + getGPA();
   return s1;
}//fim do método toString
```


O "+" e o operador de concatenação de String. Ele une os dois objetos de String.

Criando o Método Main da Classe de Driver StudentTester

- Agora crie um Método main da Classe de Driver StudentTester da seguinte forma:

```
public class StudentTester
{
   public static void main(String args[]
   {
      Student s1 = new Student(123, "Mary Smith",
                               "999-99-9999", 3.4);
      System.out.println(s1);
      Student s2 = new Student();
      s2.setStudentId(124);
      s2.setStudentName("John Jacoby");
      s2.setSSN("123-45-6789");
      s2.setGPA(4.0);
      System.out.println(s2);
      Student s3 = new Student();
      System.out.println(s3);
   }//fim do método main
}//fim da classe StudentTester
```

Blocos de Código

- Um bloco de código é definido ao abrir e fechar “chaves” { }
- Ao examinar blocos de código, considere o seguinte:
  - Cada declaração de classe é colocada em um bloco de código
  - Declarações de métodos, incluindo o método main, são colocadas em blocos de código
  - Os campos e métodos Java têm escopo (ou classe) de blocos

```
public class SayHello{                  //Bloco de Código da Classe
   public static void main(String args[]{/*Bloco de Código
                                           do Método block*/
       System.out.println("Hello Lisa");
   }//fim do método main                //Bloco de Código do Método
}//fim da classe StudentTester  //Bloco de Código da Classe
```

Formato do Bloco de Código

- Os blocos de código começam com um **{ e terminam com um }**
- Cada vez que você começa um bloco de código é necessário dar a ele um fim
- Por exemplo:
  - Todos **{ DEVEM ter uma correspondência }**
- Blocos de código podem ser encontrados em:
  - Classes
  - Métodos
  - Condicionais (instruções if, instruções switch)
  - Loops

Terminologia

- Estes são os principais termos usados nesta aula:
  - Modificadores de acesso
  - Blocos de código
  - Constantes
  - Construtores
  - Classe de driver
  - Instruções de Importação
  - Java API
  - Java comentários
  - Palavras-chave Java
  - Concatenação com minúsculas
  - Métodos

Terminologia

- Estes são os principais termos usados nesta aula:
  - Classe de objeto
  - Pacotes
  - Parâmetros
  - Classe criada pelo programador
  - Concatenação com maiúsculas
  - Variáveis

Resumo

- Nesta aula, você deverá ter aprendido a:
  - Descrever o formato geral de um programa Java
  - Descrever a diferença entre uma classe de Objeto e uma classe de Driver
  - Acessar no mínimo duas APIs de classe Java
  - Explicar e dar exemplos de palavras-chave Java
  - Criar uma classe de Objeto
  - Criar uma classe de Driver


ORACLE
Academy