UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

DISCIPLINA INF01121 - MODELOS DE LINGUAGENS DE PROGRAMAÇÃO

Professor Leandro Krug Wives

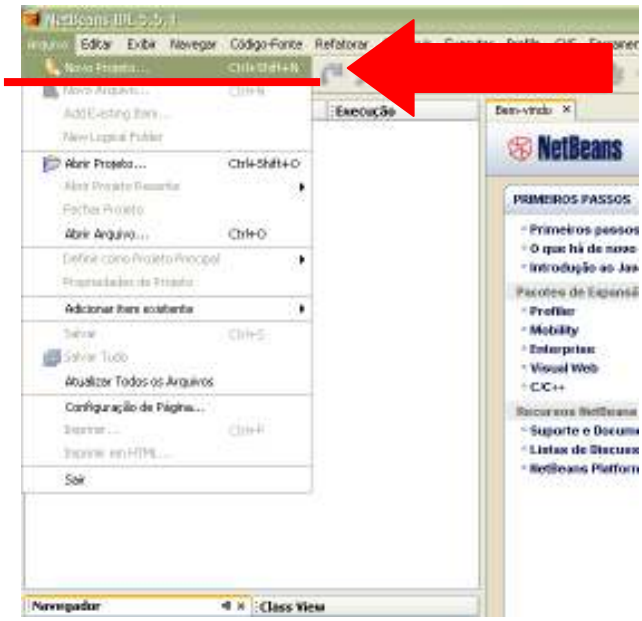
Laboratório de linguagens imperativas 01 – Introdução à linguagem Java

1) **Estudar um ambiente de desenvolvimento integrado (Eclipse, Netbeans...). Este documento é voltado ao NetBeans.**

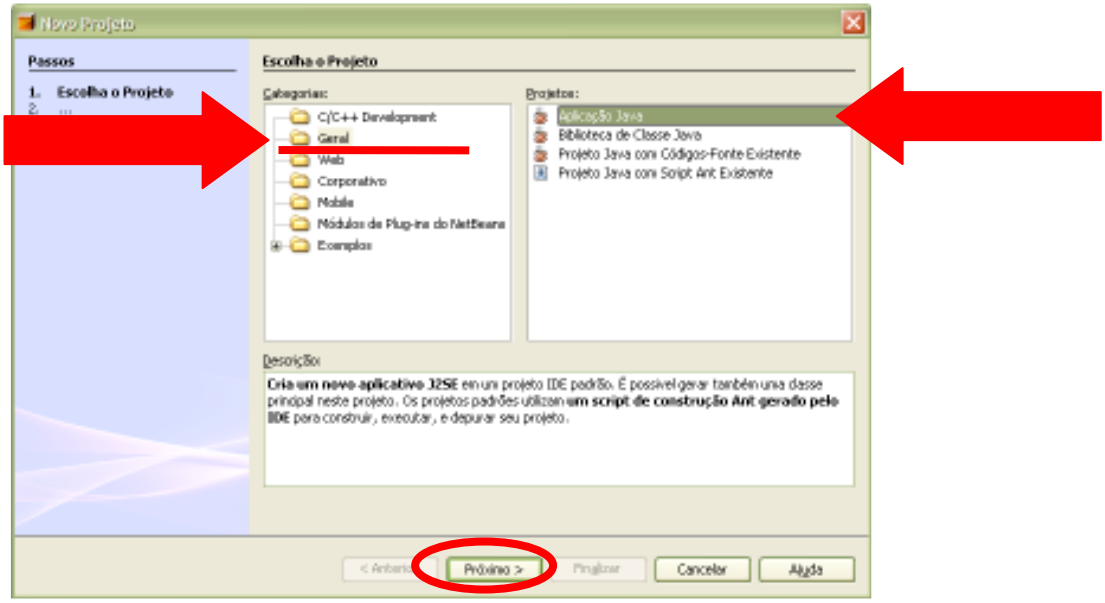
Atenção: praticamente todo ambiente trabalha com projetos. Projetos organizam o código-fonte. Os projetos encapsulam os arquivos de determinado software ou projeto (como o próprio nome diz).

a) Criando projetos com o NetBeans:

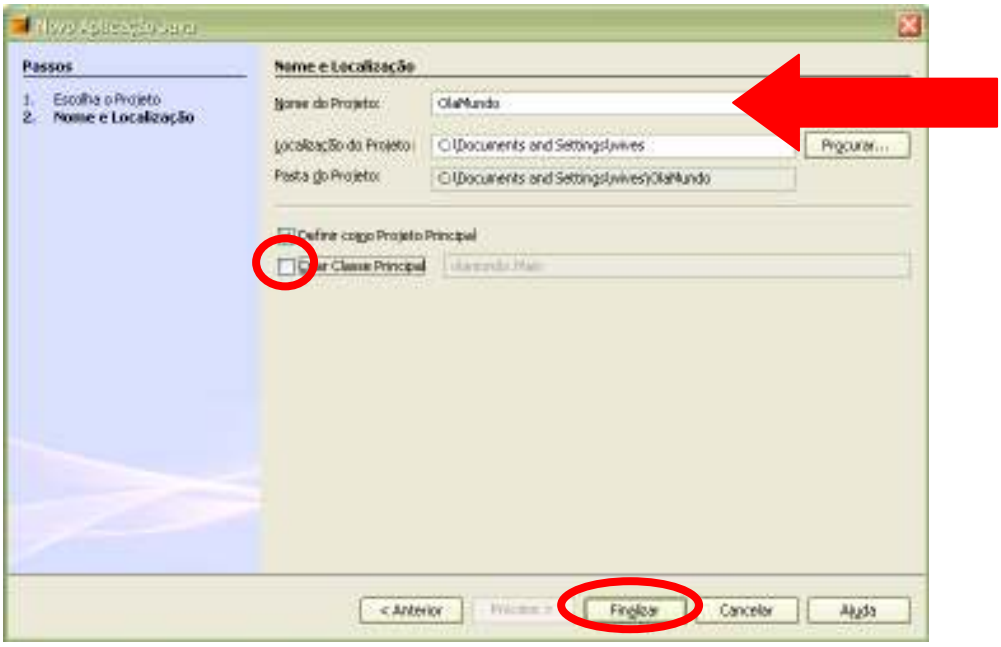
- Escolher Arquivo → Novo Projeto:



- Escolher "Aplicação Java", no item "Geral", e clicar "próximo":

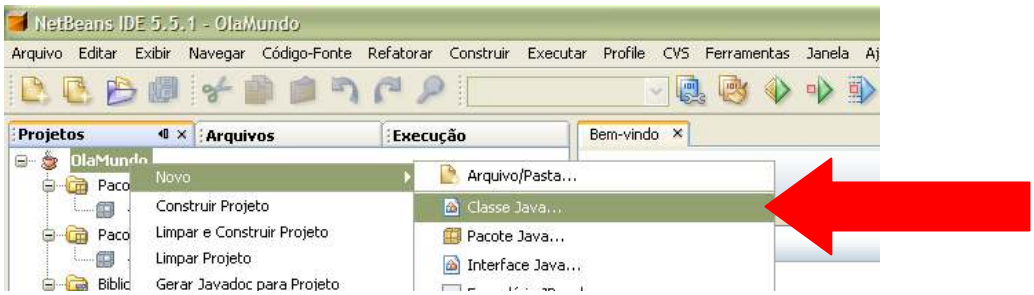


- Dar um nome para o Projeto e “Finalizar”, não se esquecendo de verificar a pasta onde o projeto será salvo e de desmarcar a opção “Criar Classe Principal”:

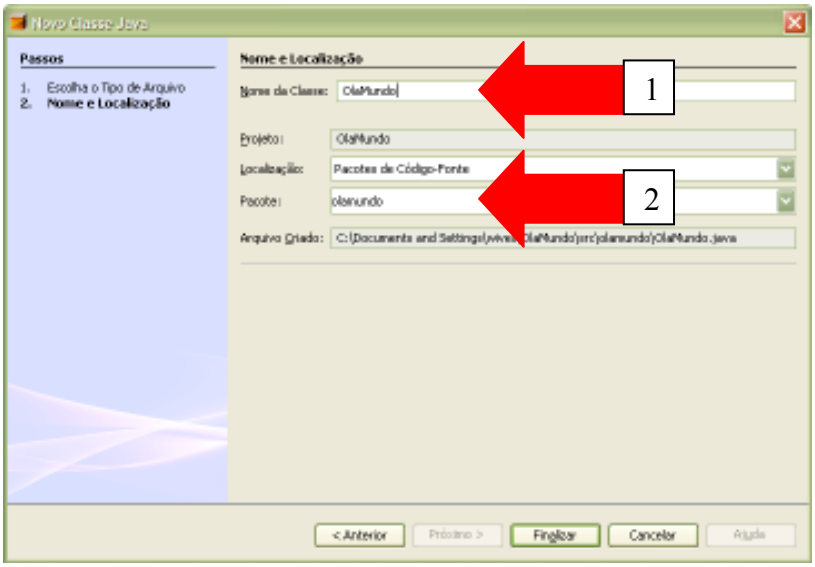


b) Adicionando classes a um projeto:

- Selecionar o projeto e clicar com botão direito nele. Escolher “Novo” → “Classe Java...”:



- Escolher o (1) nome da classe e o (2) pacote onde ela será colocada:



OBS:

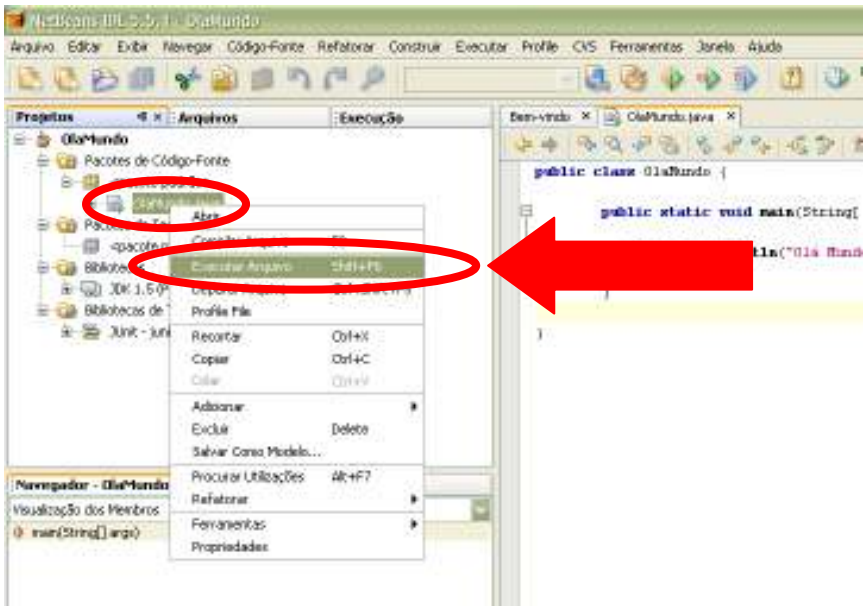
- por padrão, classes iniciam com letras maiúsculas. Pacotes são todos com letras minúsculas.

c) Executando um programa:

- Insira o código-fonte seguinte na classe “OlaMundo”, criada no passo anterior (substitua o que está lá pelo texto abaixo). Salve o arquivo.

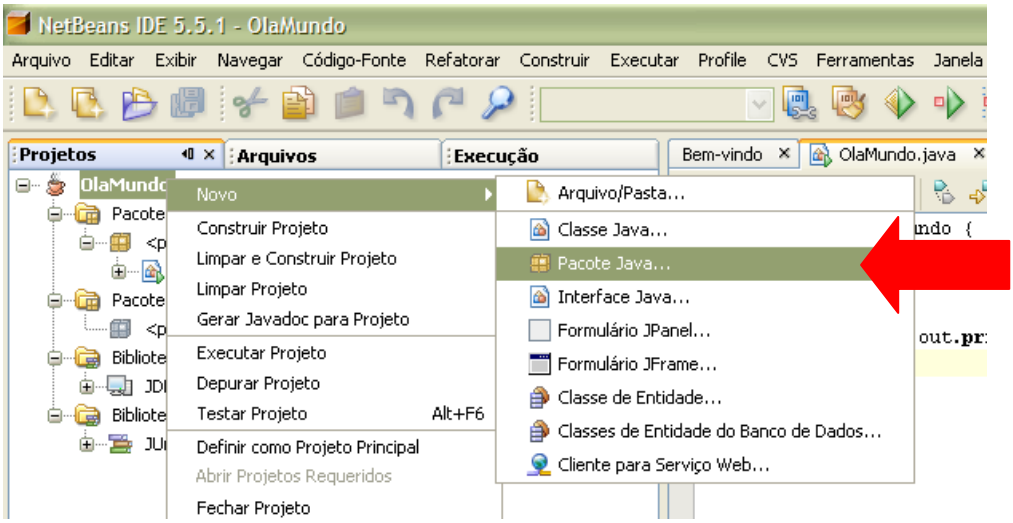
```
public class OlaMundo {
        public static void main(String[] args) {
            System.out.println("Olá Mundo!");
        }
}
```

- Selecione o projeto ou programa a ser executado e com o botão direito escolha a opção “Executar arquivo” (SHIFT + F6 também executa o arquivo atual e F6 o arquivo principal do projeto atual).



2) Criando pacotes no NetBeans

a) Selecione um projeto e com o botão direito selecione: Novo → “Pacote Java...”. Após, dê um nome ao pacote (para os exemplos seguintes, vamos precisar do pacote “pessoa”).



OBS: leia o texto no Moodle sobre pacotes.

3) Crie um projeto denominado Pessoas, contendo um pacote denominado "pessoa".

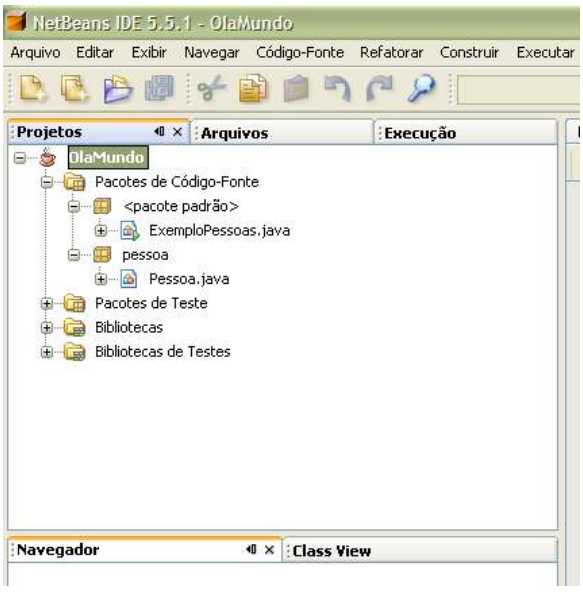
a) Dentro do pacote "pessoa", crie uma classe denominada Pessoa, para armazenar informações sobre pessoas:

```
package pessoa;

public class Pessoa {
      public String nome;
      public Integer idade;
      public String sexo;
}
```

b) Fora do pacote, mas dentro do projeto, crie uma classe para executar o programa. O programa deve criar um objeto do tipo Pessoa, atribuir valores iniciais e depois mostrar os valores:

```
import pessoa.Pessoa; // Pessoa é uma classe que está no pacote pessoa

public class ExemploPessoas {

   public static void main(String[] args) {
      Pessoa p = new Pessoa(); // new cria objetos
      // o objeto foi colocado em uma variável para posterior manipulação

      p.nome = new String("Leandro");
      p.idade = new Integer(20);
      p.sexo = new String("Masculino");

      System.out.println("Nome da pessoa: "   + p.nome);
      System.out.println("Idade da pessoa : " + p.idade);
      System.out.println("Sexo da pessoa: "   + p.sexo);

   }

}
```

OBS: A estrutura do projeto deve ser semelhante à seguinte:



c) Execute o programa ExemploPessoas.java e veja o resultado.

d) Comente as linhas que atribuem valores aos atributos de p, como no exemplo abaixo:

```
import pessoa.Pessoa; // Pessoa é uma classe que está no pacote pessoa

public class ExemploPessoas {

   public static void main(String[] args) {
      Pessoa p = new Pessoa(); // new cria objetos
      // o objeto foi colocado em uma variável para posterior manipulação
      // p.nome = new String("Leandro");
      // p.idade = new Integer(20);
      // p.sexo = new String("Masculino");
      System.out.println("Nome da pessoa: "   + p.nome);
      System.out.println("Idade da pessoa : " + p.idade);
      System.out.println("Sexo da pessoa: "   + p.sexo);

   }

}
```

➔ Execute o programa.
➔ O que aconteceu? Como fazer para atribuir valores iniciais (default) aos objetos?

4) Métodos construtores: criam objetos (podendo atribuir valores iniciais)

Quando você escreve o seguinte código:

```
Pessoa p = new Pessoa();
```

O comando new está chamando um método que possui o mesmo nome da classe Pessoa. Este método é denominado **construtor**. Mas onde ele foi definido na classe Pessoa? Não foi. A linguagem cria um método construtor padrão para qualquer classe, se o programador não definir um.

Vamos definir um método construtor para pessoa, atribuindo valores padrão. Para tanto, modifique a classe Pessoa, da seguinte forma:

```
package pessoa;

public class Pessoa {
      public String nome;
      public Integer idade;
      public String sexo;

      public Pessoa(){

            this.nome = new String("Indefinido");
            this.idade = new Integer(0);
            this.sexo = new String("Indefinido");

      }
}
```

Salve o arquivo e execute novamente o programa Pessoas. Agora a pessoa terá atributos com valores padrão!

5) Encapsulamento

- Crie uma nova classe denominada Pessoa2, dentro do pacote pessoa.

```
package pessoa;

public class Pessoa2 {
	private String nome;
	private Integer idade;
	private String sexo;

	private static int nObjPessoa = 0;

	public Pessoa2()
	{
		this.setNome("Indefinido");
		this.setIdade(0);
		this.setSexo("Indefinido");
		nObjPessoa++;
	}

	public static int nPessoas(){
		return nObjPessoa;
	}

	public void setNome(String n)
	{
		this.nome = n;
	}

	public String getNome()
	{
		return this.nome;
	}

	public void setIdade(int i)
	{
		this.idade = new Integer(i);
	}

	public void setIdade(Integer i)
	{
		this.idade = i;
	}

	public Integer getIdade()
	{
		return this.idade;
	}

	public boolean setSexo(String s)
	{
		if ( (s.equals("Masculino")) ||
		     (s.equals("masculino")) ||
		     (s.equals("Feminino"))  ||
		     (s.equals("feminino"))  ||
		     (s.equals("Indefinido"))) {

			sexo = s;
			return true;
		}
		else {
			sexo = "indefinido";
			return false;
		}
```

```
	}

	public String getSexo(){
		return sexo;
	}
}
```

Agora modifique o programa ExemploPessoas para que ele utilize esta nova classe:

```
import pessoa.Pessoa2;

public class ExemploPessoas {

	public static void main(String[] args) {
		Pessoa2 p = new Pessoa2();

		p.nome = new String("Leandro");
		p.idade = new Integer(20);
		p.sexo = new String("Masculino");

		System.out.println("Nome da pessoa: "   + p.nome);
		System.out.println("Idade da pessoa : " + p.idade);
		System.out.println("Sexo da pessoa: "   + p.sexo);

	}

}
```

Ele vai acusar erros, indicando que os atributos nome, idade e sexo não são visíveis. Agora eles são protegidos e, portanto, não podem ser acessados fora da classe Pessoa2!

- Crie um novo programa, denominado ExemploPessoas2:

```
import pessoa.Pessoa2;

public class ExemploPessoas2 {

	public static void main(String[] args) {
		Pessoa2 p = new Pessoa2();
		Pessoa2 p2 = new Pessoa2();
		Pessoa2 p3 = new Pessoa2();

		p.setNome("Leandro");
		p.setIdade(20);
		p.setSexo("Masculino");

		System.out.println("Nome da pessoa 1: "+ p.getNome());
		System.out.println("Idade da pessoa 1 : "+ p.getIdade());
		System.out.println("Sexo da pessoa 1: "+ p.getSexo());
		System.out.println("");
		System.out.println("Nome da pessoa 2: "+ p2.getNome());
		System.out.println("Idade da pessoa 2 : "+ p2.getIdade());
		System.out.println("Sexo da pessoa 2: "+ p2.getSexo());
		System.out.println("");
		System.out.println("Nome da pessoa 3: "+ p3.getNome());
		System.out.println("Idade da pessoa 3 : "+ p3.getIdade());
		System.out.println("Sexo da pessoa 3: "+ p3.getSexo());

		System.out.println("Quantidade de pessoas:"+Pessoa2.nPessoas());
	}

}
```

Perceba que, agora, você só pode modificar os atributos das pessoas através dos métodos set e get correspondentes. O método set de sexo ainda faz uma análise dos valores possíveis, validando a sua mudança.

Além disso, foi declarado um atributo de classe (static): nObjPessoa, que conta o número de objetos criados pela classe pessoa. Este atributo não pode ser manipulado pelos objetos, mas sim somente pelo método de classe nPessoas.

Exercício: modifique os métodos set dos outros atributos de forma a não aceitar nomes com números e idades entre 1 e 140 anos.

OBS:

Algumas normas:

- Nome de classes: Iniciam com letra maiúscula
- Nome de métodos: Iniciam com letra minúscula
- Nome de variáveis/objetos: Em minúsculo
- Nome de constantes: Em maiúsculo

6) Baixe o arquivo labjava01.zip. Todos os programas lá contidos devem ser colocados dentro do pacote 'olamundo'. Execute-os conforme as instruções abaixo.

a) ./olamundo/OlaMundoComString.java
Observe a instanciação de objetos. Perceba que é possível criar objetos sem associá-los às variáveis (rótulos). Neste caso, perde-se a referência ao objeto.

EXERCÍCIO: modificar o programa para que as Strings 'a' e 'b' sejam apresentadas na mesma linha.

EXERCÍCIO: modificar o argumento da instrução `new String("Olá Mundo 2!")` para `new String(a)`

b) ./olamundo/OlaMundoComMetodo.java

Observar a assinatura dos métodos.

Assinatura: <NomeClasse>.<nomeMetodo>
Assinatura de métodos non-static: <NomeObjeto>.<nomeMetodo>
OBS:
- Métodos non-static atuam sobre a memória individual de cada objeto.
- Métodos estáticos não precisam de instâncias das classes ao qual pertencem

EXERCÍCIO: fazer a linha nula ser impressa através de um método static chamado 'imprimirLinhaNula()'.

c) ./olamundo/OlaMundoComNumeros.java

Observar a representação dos dados:

Número representado como String
Número representado como tipo primitivo 'int'
Número representado como Integer

EXERCÍCIO: descomentar o bloco de instruções referente ao TESTE A e verificar o funcionamento da conversão implícita (somente nas versões acima da 1.5)
OBS: a maneira mais segura e recomendada é a explícita!

EXERCÍCIO: Descomentar o bloco de instruções referente à TESTE B
OBS:

- Método 'objeto.valueOf(x)' presente na API das principais classes
- Vantagem: converte de modo seguro o valor passado como argumento 'x' para o tipo(classe) do 'objeto'.
- Em TESTE B o tipo primitivo 'int' é convertido em Integer

EXERCÍCIO: descomentar o bloco de instruções referente ao TESTE C

Erro: Um rótulo para objetos Integer não pode receber um objeto String
Solução: utilizar o método valueOf():

```
identificador_c = Integer.valueOf(identificador_a);
```

EXERCÍCIO: descomentar o bloco de instruções referente ao TESTE D

- Verificar semelhança com o TESTE A
- Atentar que 'identificador_c' é um Integer
- Descobrir os tipos dos objetos com o método 'getClass()'
- Descobrir como usar o operador 'instanceOf' em cláusulas "if" para verificar a classe de um objeto.

OBS: Método toString() é automático e qualquer classe pode implementar (todas as classes são derivadas de Object, que o possui).

d) ./olamundo/Mundo.java

Exercício: implementar método toString!
OBS:

- Toda requisição de apresentação do objeto como uma String seguirá o padrão implementado "Planeta <NOME>".
- main() é responsável pela saudação e criação dos mundos

EXERCÍCIO: alterar o exemplo para que cada Mundo armazene, além de seu próprio nome, o diâmetro do planeta, que deverá ser informado no momento da criação do objeto (método construtor com 2 argumentos).

EXERCÍCIO: a saudação dos mundos criados deve ser modificada para que apresente o diâmetro do planeta - utilizar 'getDiametro()' para impressão!

EXERCÍCIO 13: controlar a quantidade de objetos criados com o atributo 'totalDeMundos'.
OBS:

- Sempre que um mundo for criado (método construtor chamado), a variável deverá ser incrementada.
- Ao final da execução da rotina 'main()', exibir a quantidade de mundos presentes no sistema.
- atributos static são úteis para saber quantidade de threads sendo executadas, definir chaves primárias, seqüências, etc.

e) ./olamundo/SistemaSolar.java

OBS:
- utiliza a classe Mundo definida anteriormente.
- Não há necessidade de fazer 'import', pois pertencem ao mesmo pacote com visibilidade publica.
- Um sistema solar é composto por uma lista de mundos/planetas.

Observar o uso da classe ArrayList() da API java (Opcional: LinkedList() → lista encadeada, com maior performance em determinadas operações de inserção e remoção no meio da lista).

Observe: 'import java.util.ArrayList'.

Métodos de ArrayList():
- add()
- get()
- remove()

Observe os diferentes tipos de notação:
- Outline com referencia:

```
Mundo m1 = new Mundo("Mercurio");
mundos.add(m1);
```

- Inline sem referencia:

```
mundos.add(new Mundo("Terra"));
```

Observe inserção em array já populado, em posição específica:

```
sisSol.getMundos().add(1, new Mundo("Venus"));
```

Observe remoção com índice conhecido

```
sisSol.getMundos().remove(8);    (remove plutão)
```

Observe como trabalhar com índice desconhecido, porém com objeto conhecido:

```
sisSol.getMundos().remove(plutao);
```

OBS: 'sisSol.getMundos().remove(new Mundo("plutao"));' não funcionária, pois apesar de representar a mesma coisa, são objetos distintos.

EXERCÍCIO: Baseado nos exemplos demonstrados com SistemaSolar e Mundo, criar um novo pacote 'faculdade' para representar uma TurmaDeAlunos e Aluno.

OBS:
- Cada aluno deverá possuir um nome e um número de matricula.
- A TurmaDeAlunos armazenará uma lista com pelo menos 5 alunos matriculados.
- Depois de criar uma turma, o programa deverá exibir o nome e o numero de matricula dos 5 alunos.