Forma de entrega: mostrar para o professor na aula do dia **04/abr**. Os testes deverão estar rodando no Eclipse e os membros do grupo serão indagados sobre os testes implementados.

A entrega pode ser em dupla.

Observações:

- Todas as classe de programação deverão estar no pacote aula do folder src e as classes de teste deverão estar no pacote aula do folder test;
- Todas as classes de teste deverão ser finalizadas com o sufixo Test, por exemplo, SenaTest.

Exercício 1 – Programar os seguintes testes para a interface Sena da Figura 1.

a. Simular o teste do método create para os valores 5, 6, 12 e 13. Para fazer o teste, basta verificar se o método retorna um array com n elementos;
b. Testar se todos os números retornados pelo método create estão no intervalo [1,60];
c. Testar se todos os números retornados pelo método create estão ordenados;
d. Testar se existem números repetidos no array retornado pelo método create.

Observações:

- A interface Sena **não** pode ser alterada;
- Para criar um array e preencher use a instrução: new int[]{1,2,3,4,5,6}.

```
package aula;
public interface Sena {
  /* retorna um array com valores no intervalo [1,60]
   * se n estiver no intervalo [6,12] ou uma exceção caso contrário */
  public int[] create(int n) throws NullPointerException;
}
```

Figura 1 – Código da interface Sena.

Exercício 2 – Programar os seguintes testes para a classe Cadastro da Figura 2.

a. Simular o teste do método insert para os seguintes parâmetros:

   1 e "De volta para o futuro I";

   2 e null;

   3 e "".

b. Simular o teste do método imprimir para lançar uma exceção.

Observação: a classe Cadastro **não** pode ser alterada.

```
public class Cadastro {
  private File file;
  private FileWriter fw;

  public Cadastro(String filename) throws IOException{
    file = new File(filename);
    fw = new FileWriter(file);
    fw.close();
  }
```

```
  public boolean insert(int idFilme, String nome) throws IOException{
    if( idFilme > 0 && nome != null && !nome.isEmpty() ){
      fw = new FileWriter(file,true);
      fw.write( idFilme +";"+ nome +"\r\n");
      fw.close();
      return true;
    }
    else{
      return false;
    }
  }

  public void imprimir() throws IOException{
    FileReader fr = new FileReader(file);
    BufferedReader br = new BufferedReader(fr);
    String linha = null;
    do{
      linha = br.readLine();
      if( linha != null ){
        System.out.println( linha );
      }
    }while( linha != null );
    br.close();
    fr.close();
  }
}
```

Figura 2 – Código da classe Cadastro.

Exercício 3 – Programar os seguintes testes para a interface Celula da Figura 3.

a. Simular o teste do método getNome;

b. Simular o teste do método getProxima;

c. Simular o teste do método getAnterior.

Observação: a interface Celula **não** pode ser alterada.

```
package aula;
public interface Celula {
  String nome = "";
  Celula anterior = null, proxima = null;

  public void setAnterior( Celula anterior );

  public void setProxima(Celula proxima);

  public Celula getAnterior();

  public Celula getProxima();

  public String getNome();

  public void setNome();
}
```

Figura 3 – Código da interface Celula.

Exercício 4 – Simular testes para o método get da interface Lista da Figura 4.

Observação: a interface Lista **não** pode ser alterada.

```
package aula;
public interface Lista {
  public void add(String nome );

   /* retorna o nome que está na posição n
    * ou lança uma exceção caso a posição não exista */
  public String get(int n) throws ArrayIndexOutOfBoundsException;
}
```

Figura 4 – Código da interface Lista.

Exercício 5 – Programar uma classe para simular chamadas consecutivas no método get da interface Lista da Figura 4. Simular com os seguintes valores:

- "aaa" para a 1ª chamada;
- "bbb" para a 2ª chamada;
- ArrayIndexOutOfBoundsException para a 3ª chamada.