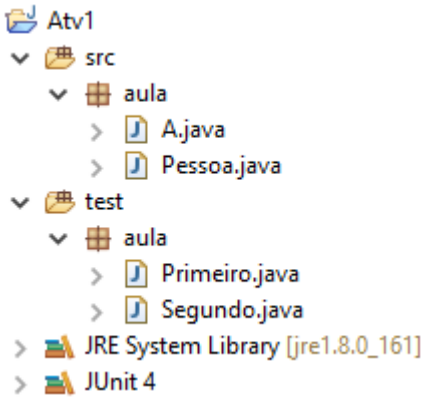
Instruções para a entrega: fazer os exercícios e entregar as respostas para o professor no início da aula do dia **01/mar**. As respostas deverão estar no papel, pode ser impresso ou a caneta (não serão aceitas respostas na tela do computador). A entrega pode ser em dupla. Alunos ausentes não terão a nota considerada.

Observações:

- Crie um projeto e adicione a biblioteca JUnit 4;
- Copie as classes a seguir de modo que o projeto fique com a estrutura representada na figura a seguir.




```java
package aula;

public class Pessoa {
    public String nome;

    @Override
    public boolean equals(Object o) {
       Pessoa p = (Pessoa) o;
        return p.nome == this.nome;
    }
}
```

```java
package aula;

public class A {
   public boolean m(Integer a){
      return a/2 < 5? true : false;
   }

   public int n(int a, int b) throws Exception{
      if( a < b ){
         return a;
      }
      else if( a > b ){
         return b;
      }
      throw new Exception();
   }

   public String o(){
      return null;
   }

   public String p(){
      return "oi";
   }

   public Pessoa q(){
      Pessoa p = new Pessoa();
      p.nome = "Ana";
      return p;
   }
}
```

```java
package aula;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Ignore;
import org.junit.Test;

public class Primeiro {
   @After
   public void a() {
      System.out.println("a");
   }

   @Before
   public void b() {
      System.out.println("b");
   }

   @AfterClass
   public static void c(){
      System.out.println("c");
   }

   @BeforeClass
   public static void d(){
      System.out.println("d");
   }

   @Ignore
   @Test
   public void e() {
      System.out.println("e");
   }

   @Test
   public void f() {
      System.out.println("f");
   }

   @Test
   public void g() {
      System.out.println("g");
   }
}
```

```java
package aula;

import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

public class Segundo {
   private A a;

   @Before
   public void setUp() {
      a = new A();
   }

   @Test
   public void a() {
      assertFalse(a.m(14));
   }

   @Test(expected=Exception.class)
   public void b() {
      assertFalse(a.m(14));
   }

   @Test(expected=Exception.class)
   public void c() {
      assertFalse(a.m(null));
   }

   @Test
   public void d() throws Exception {
      assertEquals(1, a.n(1, 2), 0);
   }

   @Test
   public void e() throws Exception {
      assertEquals(2, a.n(2, 2), 0);
   }

   @Test
   public void f() {
      Pessoa p = new Pessoa();
      p.nome = "Ana";
      assertEquals(p, a.q());
   }

   @Test
   public void g() {
      Pessoa p = new Pessoa();
      p.nome = "Ana";
      assertSame(p, a.q());
   }
}
```

Exercício 1 – Qual é a sequência de letras ao executar a classe de teste Primeiro?

Exercício 2 – Por que os métodos anotados com @BeforeClass e @AfterClass precisam ser estáticos?

Exercício 3 – Por que o método b(), da classe de teste Segundo, apresenta um resultado azul e o método c() um resultado verde?

Exercício 4 – O método e(), da classe de teste Segundo, apresenta um resultado vermelho. Qual é a alteração necessária no método de teste e() para que o resultado seja verde?

Exercício 5 – Programar um método de teste para testar o método o(), da classe A. O resultado do teste deverá ser verde.

Exercício 6 – Programar um método de teste para testar o método p(), da classe A. O resultado do teste deverá ser verde. Observação: use o método assertSame da classe org.junit.Assert.

Exercício 7 – Qual é o motivo dos métodos assertEquals e assertSame produzirem resultados diferentes nos teste dos métodos f() e g() da classe Segundo?