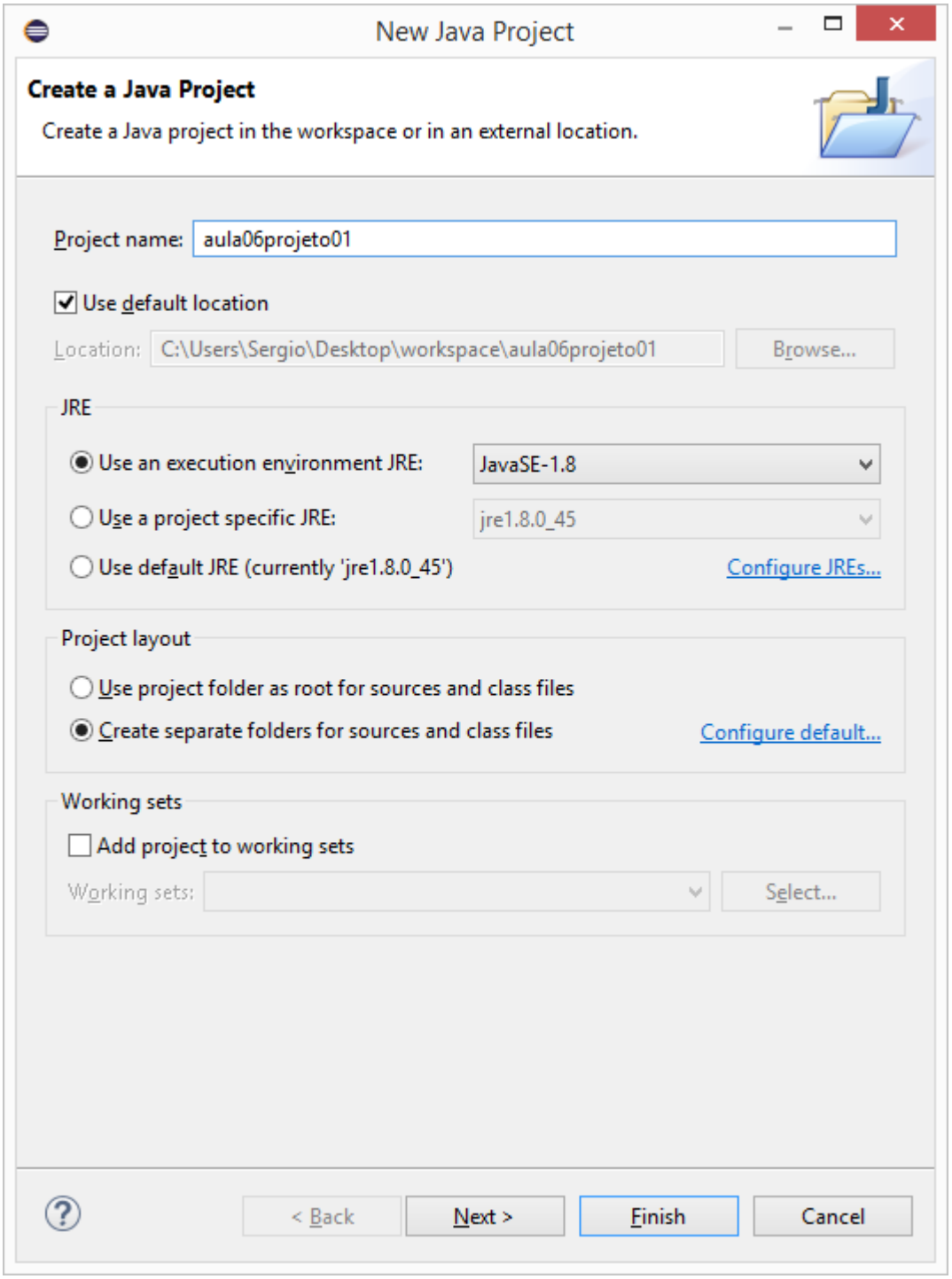
Novo projeto:

File > New > Java Project



Criando a tabela de mensagem no banco de dados:

```
drop database if exists aula06;
create database aula06;
use aula06;

create table mensagem(
	idmensagem	integer		auto_increment,
	emaildest	varchar(50)	not null,
	assunto		varchar(50)	not null,
	conteudo	varchar(255)	not null,
	dataenvio	timestamp	not null,
	primary key(idmensagem));

desc mensagem;
```

Criando a classe de entidade:

```
package entities;

import java.util.Date;

public class Mensagem {

	private Integer idMensagem;
	private String emailDest;
	private String assunto;
	private String conteudo;
	private Date dataEnvio;

	public Mensagem() {
		// TODO Auto-generated constructor stub
	}

	public Mensagem(Integer idMensagem, String emailDest,
		String assunto, String conteudo, Date dataEnvio) {
		this.idMensagem = idMensagem;
		this.emailDest = emailDest;
		this.assunto = assunto;
		this.conteudo = conteudo;
		this.dataEnvio = dataEnvio;
	}
```

```
public Integer getIdMensagem() {
	return idMensagem;
}

public void setIdMensagem(Integer idMensagem) {
	this.idMensagem = idMensagem;
}

public String getEmailDest() {
	return emailDest;
}

public void setEmailDest(String emailDest) {
	this.emailDest = emailDest;
}

public String getAssunto() {
	return assunto;
}

public void setAssunto(String assunto) {
	this.assunto = assunto;
}

public String getConteudo() {
	return conteudo;
}

public void setConteudo(String conteudo) {
	this.conteudo = conteudo;
}

public Date getDataEnvio() {
	return dataEnvio;
}

public void setDataEnvio(Date dataEnvio) {
	this.dataEnvio = dataEnvio;
}

@Override
public String toString() {
	return "Mensagem [idMensagem=" + idMensagem
	+ ", emailDest=" + emailDest + ", assunto=" + assunto
	+ ", conteudo=" + conteudo + ", dataEnvio="
	+ dataEnvio + "]";
```

```
	}
}
```

Criando a Classe para acesso ao banco de dados:

```
package persistence;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class Dao {

	private static final String DRIVER = "com.mysql.jdbc.Driver";
	private static final String URL = "jdbc:mysql://localhost:3306/aula06";
	private static final String USER = "root";
	private static final String PASSWORD = "brqbrq";

	//API do jdbc (java.sql.*)
	protected Connection con;
	protected PreparedStatement stmt;
	protected CallableStatement call;
	protected ResultSet rs;

	//métodos para acessar a base de dados..
	protected void openConnection() throws Exception{
		Class.forName(DRIVER);
		con = DriverManager.getConnection(URL, USER, PASSWORD);
	}

	protected void closeConnection() throws Exception{
		if(con != null){
			con.close();
		}
	}
}
```

---

persistence/MensagemDao.java

```
package persistence;

import java.util.ArrayList;
import java.util.List;
```

```
import entities.Mensagem;
import util.FormatacaoData;

public class MensagemDao extends Dao{

	//método para gravar uma mensagem na base de dados..
	public void insert(Mensagem msg) throws Exception{

		String query = "insert into mensagem
				(emaildest, assunto, conteudo, dataenvio) "
				 + "values(?, ?, ?, ?)";

		openConnection();
		stmt = con.prepareStatement(query);
		stmt.setString(1, msg.getEmailDest());
		stmt.setString(2, msg.getAssunto());
		stmt.setString(3, msg.getConteudo());
		stmt.setString(4, FormatacaoData.convertToString(msg.getDataEnvio()));
		stmt.execute();
		stmt.close();

		closeConnection();
	}

	//método para listar todas as mensagem cadastradas na tabela..
	public List<Mensagem> findAll() throws Exception{

		String query = "select * from mensagem order by dataenvio desc";

		openConnection();
		stmt = con.prepareStatement(query);
		rs = stmt.executeQuery();

		List<Mensagem> lista = new ArrayList<Mensagem>();

		while(rs.next()){
			Mensagem msg = new Mensagem(); //entidade..
			msg.setIdMensagem(rs.getInt("idmensagem"));
			msg.setEmailDest(rs.getString("emaildest"));
			msg.setAssunto(rs.getString("assunto"));
			msg.setConteudo(rs.getString("conteudo"));
			msg.setDataEnvio(FormatacaoData
				.convertToDate(rs.getString("dataenvio")));

			lista.add(msg);
		}
```

```
            stmt.close();
            closeConnection();

            return lista; //retornando a lista..
        }

}
```

Classe auxiliar para conversão da data do mysql

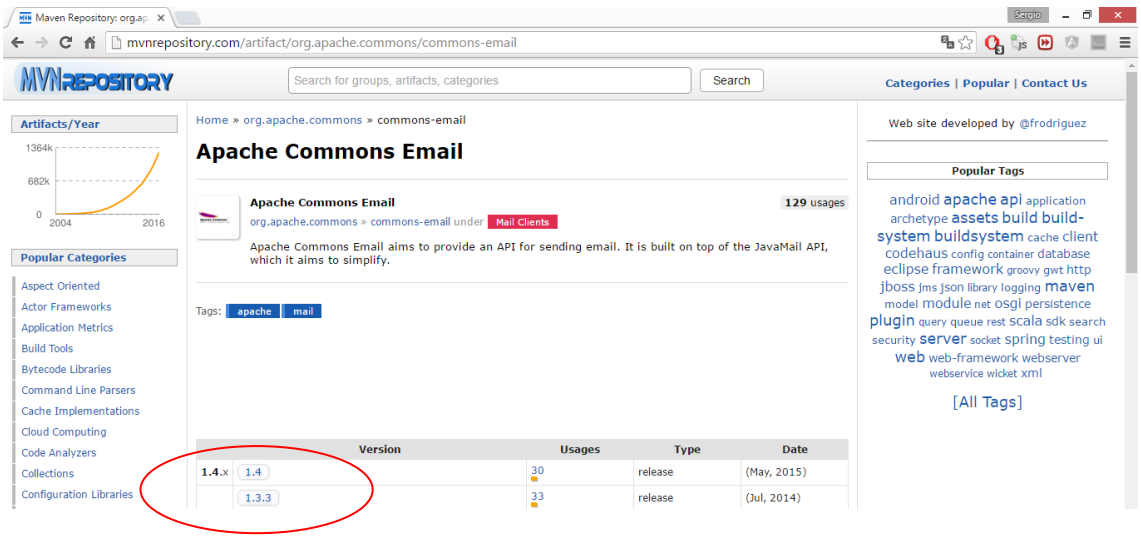
```
package util;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

public class FormatacaoData {

        //método estatico para receber uma data do java e retorna-la
        //como string no padrão do mysql (yyyy-MM-yy HH:mm:ss)
        public static String convertToString(Date data){

            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
            return sdf.format(data);
        }

        //método estatico para receber uma string e retorna-la como Date
        public static Date convertToDate(String data){

            int ano = Integer.parseInt(data.substring(0,4));
            int mes = Integer.parseInt(data.substring(5,7));
            int dia = Integer.parseInt(data.substring(8,10));

            int hora = Integer.parseInt(data.substring(11,13));
            int minuto = Integer.parseInt(data.substring(14,16));
            int segundo = Integer.parseInt(data.substring(17,19));

            Calendar cal = new GregorianCalendar
                    (ano, mes-1, dia, hora, minuto, segundo);

            return cal.getTime(); //retornando o Callendar como Date..
        }

}
```

Criando a classe para leitura dos dados através do Usuario:

```
package input;

import javax.swing.JOptionPane;

//classe para ler os dados da mensagem informados pelo usuario
public class InputMensagem {

	//ler o email do destinatario
	public String lerEmailDest(){
		return JOptionPane.showInputDialog
			("Informe o email do destinatario:");
	}

	//ler o assunto da mensagem
	public String lerAssunto(){
		return JOptionPane.showInputDialog
			("Informe o assunto:");
	}

	//ler o conteudo da mensagem
	public String lerConteudo(){
		return JOptionPane.showInputDialog
			("Conteudo da mensagem:");
	}
}
```

Apache Commons Email

Biblioteca para envio de emails
http://mvnrepository.com/artifact/org.apache.commons/commons-email



Classe para implementação do serviço de encio de emails:

```
package servivces;

import org.apache.commons.mail.Email;
import org.apache.commons.mail.SimpleEmail;

import entities.Mensagem;

//classe para serviços de envio de email..
public class EmailService {

	//parametros de configuração (constantes)
	private static final String CONTA = "cotiexemplo@gmail.com";
	private static final String SENHA = "@coticoti@";
	private static final String SMTP = "smtp.gmail.com";
	private static final Integer PORTA = 465;

	//método para enviar email..
	public void EnviarEmail(Mensagem msg) throws Exception{

		//configurar o envio do email..
		Email e = new SimpleEmail();

		e.setHostName(SMTP); //protocolo de envio de email.
		e.setSmtpPort(PORTA); //porta para envio de email
		e.setAuthentication(CONTA, SENHA); //conta que fara o envio
		e.setTLS(true); //segurança..
		e.setSSL(true); //segurança..
		e.setFrom(CONTA, "COTI INFORMATICA"); //remetente

		e.addTo(msg.getEmailDest()); //destinatario do email..
		e.setSubject(msg.getAssunto()); //assunto do email..
		e.setMsg(msg.getConteudo()); //conteudo da mensagem..

		//enviar o email..
		e.send(); //disparando o email..
	}
}
```

Interface de controle:

```
package contracts;

//programa abstrato..
public interface IControleMensagem {
```

```
	//métodos publicos e abstratos..

	void EnviarMensagem();

	void ImprimirMensagens();

}
```

Implementando a interface:

```
package control;

import java.util.Date;

import contracts.IControleMensagem;
import entities.Mensagem;
import input.InputMensagem;
import persistence.MensagemDao;
import servivces.EmailService;

public class ControleMensagem implements IControleMensagem{

	//classe de leitura de dados..
	private InputMensagem input; //null

	//construtor..
	public ControleMensagem() {
		input = new InputMensagem(); //instanciando..
	}

	@Override
	public void EnviarMensagem() {

		try{
			Mensagem msg = new Mensagem();

			msg.setEmailDest(input.lerEmailDest());
			msg.setAssunto(input.lerAssunto());
			msg.setConteudo(input.lerConteudo());
			msg.setDataEnvio(new Date()); //data atual..

			//enviar a mensagem..
			EmailService service = new EmailService();
			//classe para envio da mensagem...
			service.EnviarEmail(msg); //executando o envio..
```

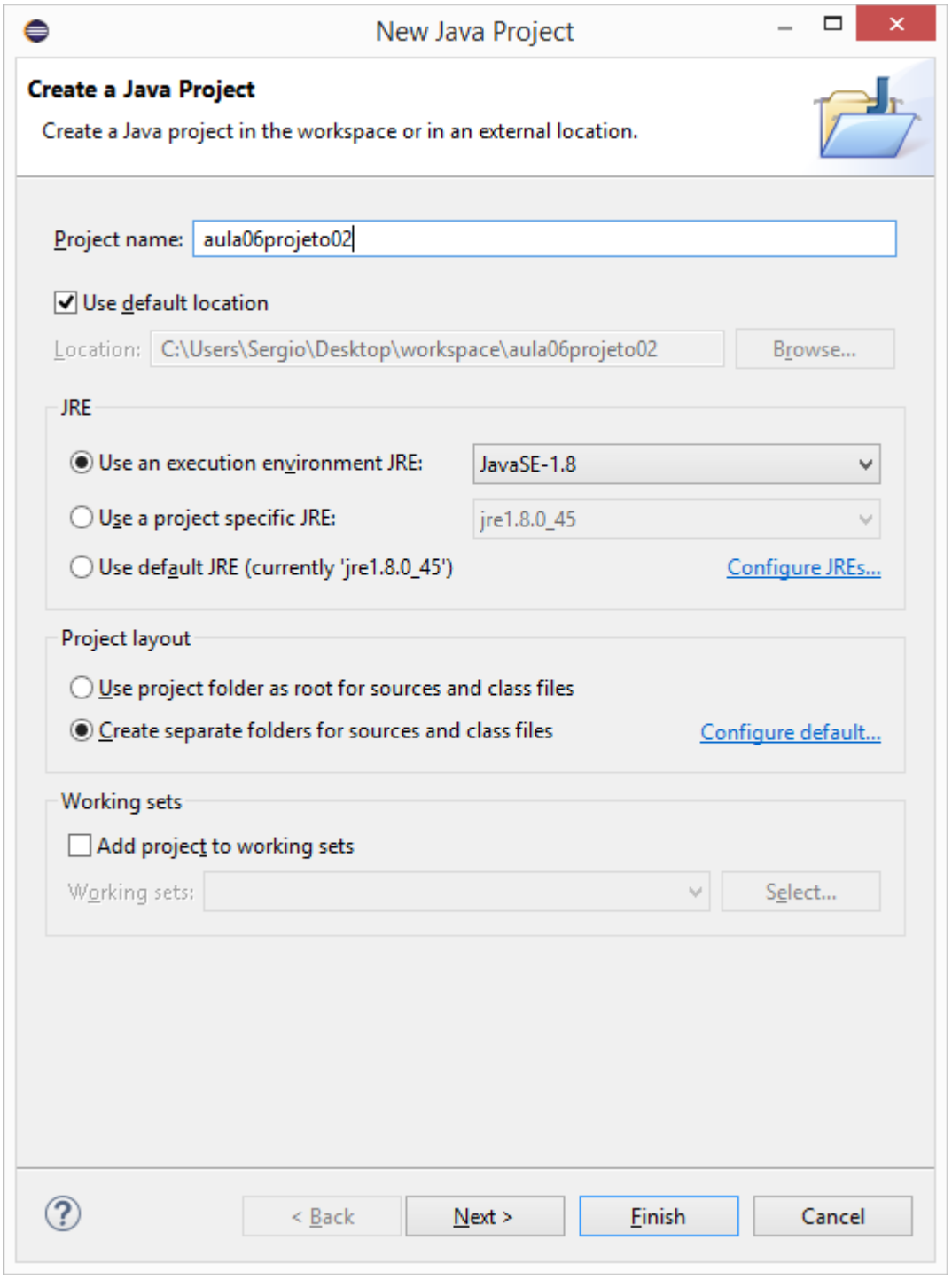
```
                //gravar na base de dados..
                MensagemDao d = new MensagemDao(); //classe de persistencia..
                d.insert(msg); //gravando no banco de dados..

                System.out.println("Mensagem enviada com sucesso.");
            }
            catch(Exception e){
                //imprimir mensagem de erro..
                System.out.println("Erro: " + e.getMessage());
            }

        }

        @Override
        public void ImprimirMensagens() {

            try{

                //classe de persistencia..
                MensagemDao d = new MensagemDao();

                System.out.println("\nListagem de mensagens enviadas:");
                for(Mensagem msg : d.findAll()){
                    //imrpimindo o objeto mensagem..
                    System.out.println(msg); //ToString();
                }
            }
            catch(Exception e){
                //imprimir mensagem de erro..
                System.out.println("Erro:" + e.getMessage());
            }
        }
}
```

Executando:

```
package principal;
import control.ControleMensagem;

public class Main {

      public static void main(String[] args) {
            ControleMensagem c = new ControleMensagem();
            c.EnviarMensagem();
            c.ImprimirMensagens();
      }
}
```

Novo Projeto:

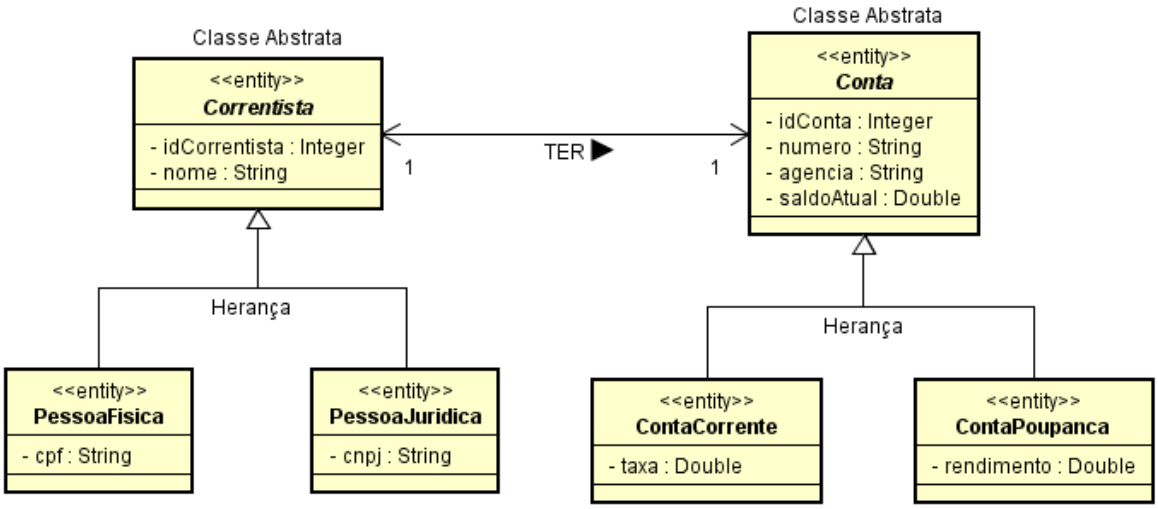
File > New > Java Project



Classes Abstratas

As classes abstratas são as que não permitem realizar qualquer tipo de instância. São classes feitas especialmente para serem modelos para suas classes derivadas. As classes derivadas, via de regra, deverão **sobrescrever os métodos** para realizar a implementação dos mesmos. As classes derivadas das classes abstratas são conhecidas como classes concretas.

Modelagem de entidades:




```
package entities;

//Classe abstrata..
public abstract class Conta {

	// atributo..
	private Integer idConta;
	private String numero;
	private String agencia;
	private Double saldoAtual;
	private Correntista correntista;

	// construtor default..
	public Conta() {
		// TODO Auto-generated constructor stub
	}

	// sobrecarga de construtores..
	public Conta(Integer idConta, String numero,
			String agencia, Double saldoAtual) {
		this.idConta = idConta;
		this.numero = numero;
```

```
		this.agencia = agencia;
		this.saldoAtual = saldoAtual;
	}

	// sobrecarga de construtores..
	public Conta(Integer idConta, String numero,
			String agencia, Double saldoAtual,
			Correntista correntista) {
		this(idConta, numero, agencia, saldoAtual);
		this.correntista = correntista;
	}

	public Integer getIdConta() {
		return idConta;
	}

	public void setIdConta(Integer idConta) {
		this.idConta = idConta;
	}

	public String getNumero() {
		return numero;
	}

	public void setNumero(String numero) {
		this.numero = numero;
	}

	public String getAgencia() {
		return agencia;
	}

	public void setAgencia(String agencia) {
		this.agencia = agencia;
	}

	public Double getSaldoAtual() {
		return saldoAtual;
	}

	public void setSaldoAtual(Double saldoAtual) {
		this.saldoAtual = saldoAtual;
	}

	public Correntista getCorrentista() {
		return correntista;
	}
```

```
	public void setCorrentista(Correntista correntista) {
		this.correntista = correntista;
	}

	@Override
	public String toString() {
		return "Conta [idConta=" + idConta + ", numero="
			+ numero + ", agencia=" + agencia + ", saldoAtual="
			+ saldoAtual + "]";
	}
}


package entities;

public class ContaCorrente extends Conta{

	private Double taxa;

	public ContaCorrente() {
		// TODO Auto-generated constructor stub
	}

	public ContaCorrente(Integer idConta, String numero, String
agencia, Double saldoAtual, Double taxa) {
		super(idConta, numero, agencia, saldoAtual);
		this.taxa = taxa;
	}

	public ContaCorrente(Integer idConta, String numero, String
agencia, Double saldoAtual, Correntista correntista,
			Double taxa) {
		super(idConta, numero, agencia, saldoAtual, correntista);
		this.taxa = taxa;
	}

	public Double getTaxa() {
		return taxa;
	}

	public void setTaxa(Double taxa) {
		this.taxa = taxa;
	}

	@Override
	public String toString() {
		return super.toString() + ", taxa: " + taxa;
	}

}
```

```java
package entities;

public class ContaPoupanca extends Conta{

	private Double rendimento;

	public ContaPoupanca() {
		// TODO Auto-generated constructor stub
	}

	public ContaPoupanca(Integer idConta, String numero,
		String agencia, Double saldoAtual, Double rendimento) {
		super(idConta, numero, agencia, saldoAtual);
		this.rendimento = rendimento;
	}

	public ContaPoupanca(Integer idConta, String numero,
			String agencia, Double saldoAtual,
			Correntista correntista,
			Double rendimento) {
		super(idConta, numero, agencia, saldoAtual, correntista);
		this.rendimento = rendimento;
	}

	public Double getRendimento() {
		return rendimento;
	}

	public void setRendimento(Double rendimento) {
		this.rendimento = rendimento;
	}

	@Override
	public String toString() {
		return super.toString() + ", rendimento: " + rendimento;
	}

}
```

```java
package entities;

//Correntista é uma Classe Abstrata..
public abstract class Correntista {

	// atributos..
	private Integer idCorrentista;
	private String nome;
	private Conta conta; // Associação (TER-1)
```

```
	// construtor vazio..
	public Correntista() {
		// TODO Auto-generated constructor stub
	}

	// sobrecarga de construtores (passagem de argumentos)
	public Correntista(Integer idCorrentista, String nome) {
		this.idCorrentista = idCorrentista;
		this.nome = nome;
	}

	// sobrecarga de construtores (passagem de argumentos)
	public Correntista(Integer idCorrentista, String nome,
			Conta conta) {
		this(idCorrentista, nome);
		this.conta = conta;
	}

	public Integer getIdCorrentista() {
		return idCorrentista;
	}

	public void setIdCorrentista(Integer idCorrentista) {
		this.idCorrentista = idCorrentista;
	}

	public String getNome() {
		return nome;
	}

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

	public Conta getConta() {
		return conta;
	}

	public void setConta(Conta conta) {
		this.conta = conta;
	}

	@Override
	public String toString() {
		return "Correntista [idCorrentista=" + idCorrentista
			+ ", nome=" + nome + "]";
	}

}
```

```java
package entities;

public class PessoaFisica extends Correntista{

	private String cpf;

	public PessoaFisica() {
		// TODO Auto-generated constructor stub
	}

	public PessoaFisica(Integer idCorrentista, String nome,
			String cpf) {
		super(idCorrentista, nome);
		this.cpf = cpf;
	}

	public PessoaFisica(Integer idCorrentista, String nome,
			Conta conta, String cpf) {
		super(idCorrentista, nome, conta);
		this.cpf = cpf;
	}

	public String getCpf() {
		return cpf;
	}

	public void setCpf(String cpf) {
		this.cpf = cpf;
	}

	@Override
	public String toString() {
		return super.toString() + ", cpf: " + cpf;
	}

}
```

```java
package entities;

//PessoaJuridica É UM Correntista
public class PessoaJuridica extends Correntista{

	private String cnpj;

	public PessoaJuridica() {
		// TODO Auto-generated constructor stub
	}
```

```
	public PessoaJuridica(Integer idCorrentista, String nome,
				String cnpj) {
		super(idCorrentista, nome);
		this.cnpj = cnpj;
	}

	public PessoaJuridica(Integer idCorrentista, String nome,
				Conta conta, String cnpj) {
		super(idCorrentista, nome, conta);
		this.cnpj = cnpj;
	}

	public String getCnpj() {
		return cnpj;
	}

	public void setCnpj(String cnpj) {
		this.cnpj = cnpj;
	}

	@Override
	public String toString() {
		return super.toString() + ", cnpj: " + cnpj;
	}
}
```

Classe abstrata para controle de Conta

```
package control;

import entities.Conta;

//Classe abstrata
public abstract class ControleConta<T extends Conta> {

	//Em Java, Classes abstratas podem ter construtores,
	metodos, atributos, etc..
	//Mas tambem podem ter metodos abstratos, ou seja,
	metodos que so possuam
	//assinatura e que deverão ser implementados
	pelas subclasses que herdarem
	//a classe abstrata...

	//método para realizar deposito em conta
	public abstract void realizarDeposito(T conta, Double valor);
```

```
	//método para realizar saque em conta
	public abstract void realizarSaque(T conta, Double valor)
		throws Exception;

	//método para imprimir os dados da conta..
	//final em metodos -> o metodo nao pode ser sobrescrito..
	public final void imprimirDados(T conta){

		System.out.println("\nDados da Conta:");
		System.out.println("Id da Conta.........: "
			+ conta.getIdConta());
		System.out.println("Correntista.........: "
			+ conta.getCorrentista());
		System.out.println("Numero da Conta.....: "
			+ conta.getNumero());
		System.out.println("Agencia.............: "
			+ conta.getAgencia());
		System.out.println("Saldo...............: "
			+ conta.getSaldoAtual());
	}
}
```

Herdando a Classe abstrata:

```
package control;

import entities.ContaCorrente;

public class ControleContaCorrente extends
ControleConta<ContaCorrente>{

	@Override
	public void realizarDeposito
		(ContaCorrente conta, Double valor) {

		Double saldo = conta.getSaldoAtual();
		saldo += valor; //acumulador..

		conta.setSaldoAtual(saldo); //novo saldo..
	}

	@Override
	public void realizarSaque(ContaCorrente conta, Double valor)
		throws Exception {

		Double saque = (valor * conta.getTaxa()) + valor;


		//se ha limite para saque..
```

```
            if(conta.getSaldoAtual() >= saque){

                  Double saldo = conta.getSaldoAtual();
                  conta.setSaldoAtual(saldo - saque);
            }
            else{
                  throw new Exception("Saldo insuficiente para
                                    realizar o saque.");
            }
      }
}
```

```
package control;

import entities.ContaPoupanca;

public class ControleContaPoupanca extends
ControleConta<ContaPoupanca>{

      @Override
      public void realizarDeposito
            (ContaPoupanca conta, Double valor) {

            Double deposito = (valor * conta.getRendimento()) + valor;
            Double saldo = conta.getSaldoAtual();

            conta.setSaldoAtual(saldo + deposito);
      }

      @Override
      public void realizarSaque(ContaPoupanca conta, Double valor)
            throws Exception {

            if(conta.getSaldoAtual() >= valor){
                  Double saldo = conta.getSaldoAtual();
                  conta.setSaldoAtual(saldo - valor);
            }
            else{
                  throw new Exception("Saldo insuficiente
                                    para saque.");
            }

      }

}
```

Executando:

package principal;

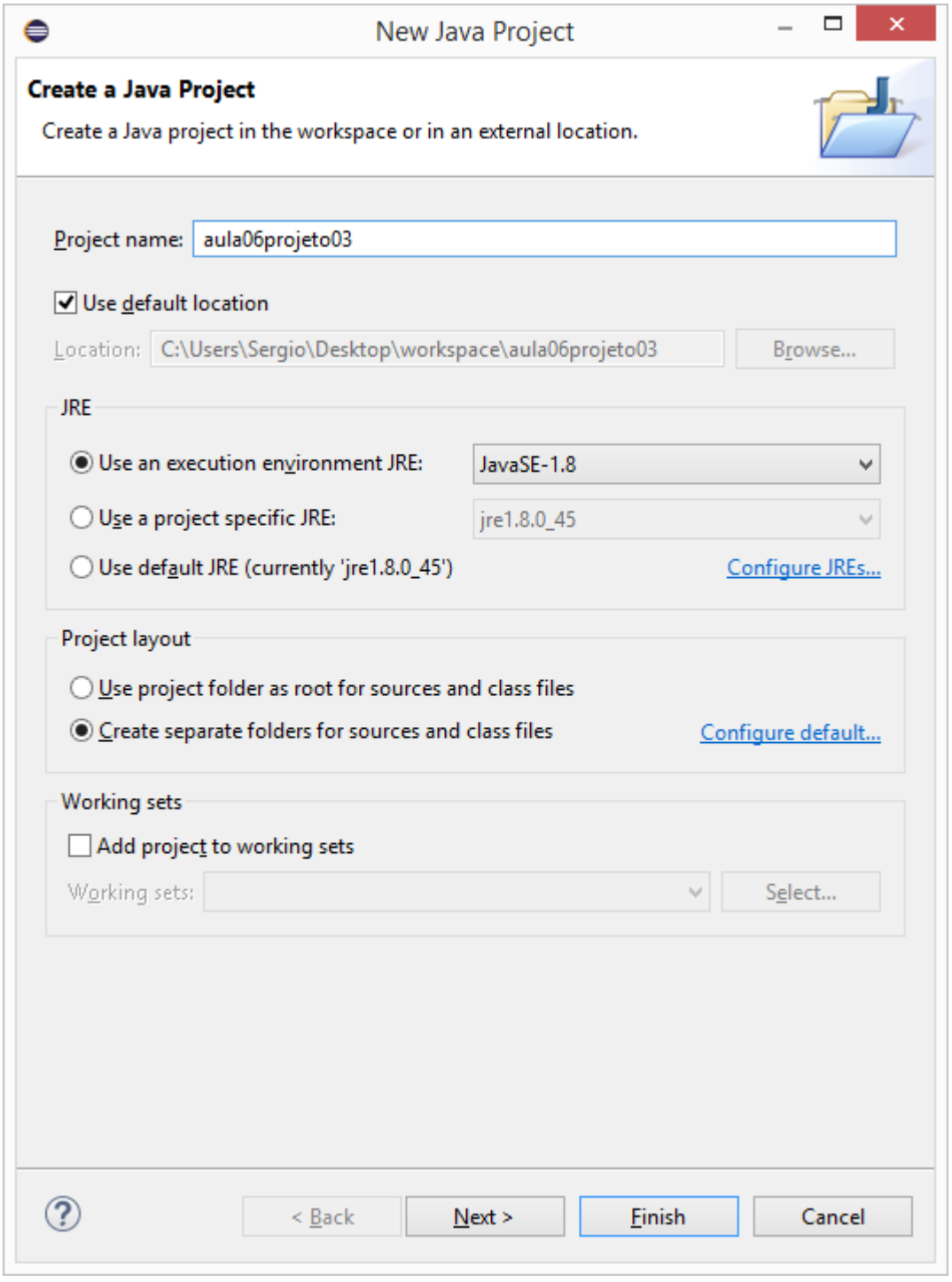
```
import control.ControleConta;
import control.ControleContaCorrente;
import entities.ContaCorrente;
import entities.PessoaFisica;

public class Main {

	public static void main(String[] args) {

		PessoaFisica pf = new PessoaFisica();
		pf.setIdCorrentista(1);
		pf.setCpf("1234567890");
		pf.setNome("Sergio Mendes");
		ContaCorrente cc = new ContaCorrente();
		cc.setIdConta(1);
		cc.setNumero("1234-5");
		cc.setAgencia("9876-0");
		cc.setSaldoAtual(0.0);
		cc.setTaxa(0.05);
		cc.setCorrentista(pf); //relacionando conta ao correntista

		ControleContaCorrente controle = new ControleContaCorrente();
		controle.realizarDeposito(cc, 1000.0);
		controle.realizarDeposito(cc, 200.0);

		try {
			controle.realizarSaque(cc, 400.0);
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
		controle.imprimirDados(cc);
	}
}
```

Saida do programa:

```
Dados da Conta:
Id da Conta..........: 1
Correntista..........: Correntista [idCorrentista=1, nome=Sergio
Mendes], cpf: 1234567890
Numero da Conta......: 1234-5
Agencia..............: 9876-0
Saldo................: 780.0
```

Novo Projeto:

File > New > Java Project



Criando uma entidade Aluno:
entities/Aluno.java

```java
package entities;

import java.util.Arrays;

public class Aluno {

	private Integer idAluno;
	private String nome;
	private Double[] notas;

	public Aluno() {
		// TODO Auto-generated constructor stub
	}

	public Aluno(Integer idAluno, String nome, Double[] notas)
	{
		this.idAluno = idAluno;
		this.nome = nome;
		this.notas = notas;
	}

	public Integer getIdAluno() {
		return idAluno;
	}

	public void setIdAluno(Integer idAluno) {
		this.idAluno = idAluno;
	}

	public String getNome() {
		return nome;
	}

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

	public Double[] getNotas() {
		return notas;
	}


	public void setNotas(Double[] notas) {
		this.notas = notas;
	}
```

```
	@Override
	public String toString() {
		return "Aluno [idAluno=" + idAluno + ", nome=" + nome
		+ ", notas=" + Arrays.toString(notas) + "]";
	}
}
```

Criando uma Classe de controle para operações com Aluno:

```
package control;

import entities.Aluno;
import types.SituacaoAluno;

public class ControleAluno {

	//método para calcular a media de um aluno baseado nas notas..
	public Double obterMedia(Aluno a) throws Exception{

		//somar as notas do aluno..
		double somatorio = 0.0;

		//varrendo cada nota contida no vetor..
		for(Double nota : a.getNotas()){
			//verificar se a nota percorrido no vetor é valida..
			if(nota >= 0 && nota <= 10){
				somatorio += nota; //acumulador..
			}
			else{
				throw new Exception("Nota invalida: " + nota);
			}
		}

		//calcular a media e retornar o valor..
		return somatorio / a.getNotas().length;
	}

	//método para retornar a situação do aluno, baseado na media..
	public SituacaoAluno obterSituacao(Aluno a) throws Exception{

		//calcular a media do aluno...
		Double media = obterMedia(a);

		if(media >= 7){
			return SituacaoAluno.Aprovado;
		}
		else if(media >= 5){
			return SituacaoAluno.Recuperacao;
```

```
            }
            else{
                  return SituacaoAluno.Reprovado;
            }
      }
}
```

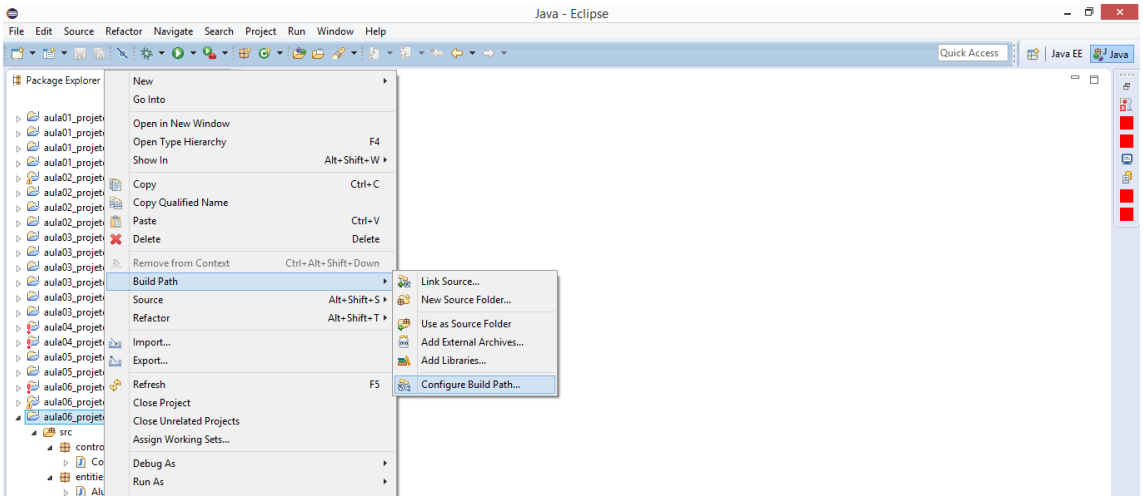
Junit

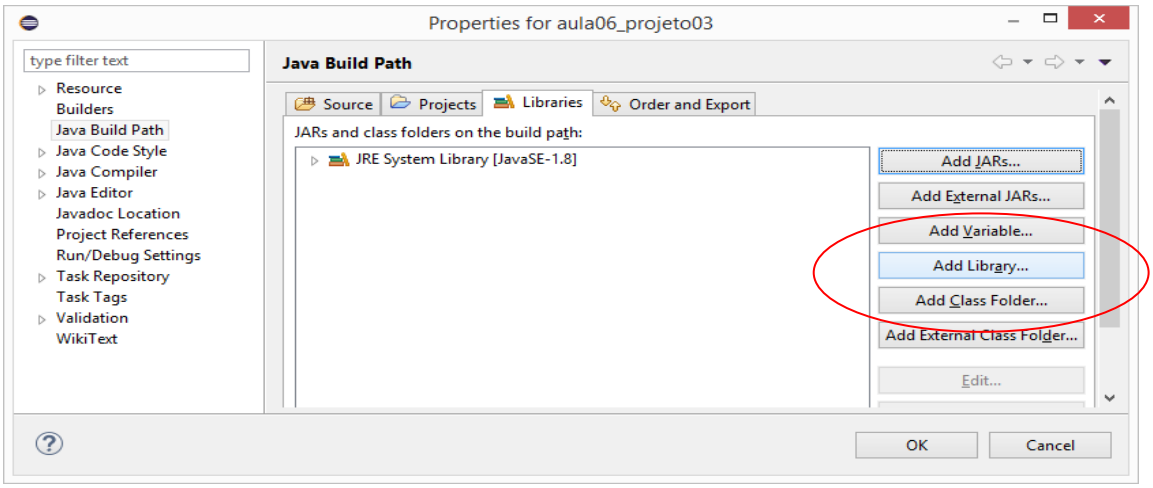
Esse framework facilita a criação de código para a automação de testes com apresentação dos resultados. Com ele, pode ser verificado se cada método de uma classe funciona da forma esperada, exibindo possíveis erros ou falhas podendo ser utilizado tanto para a execução de baterias de testes como para extensão.

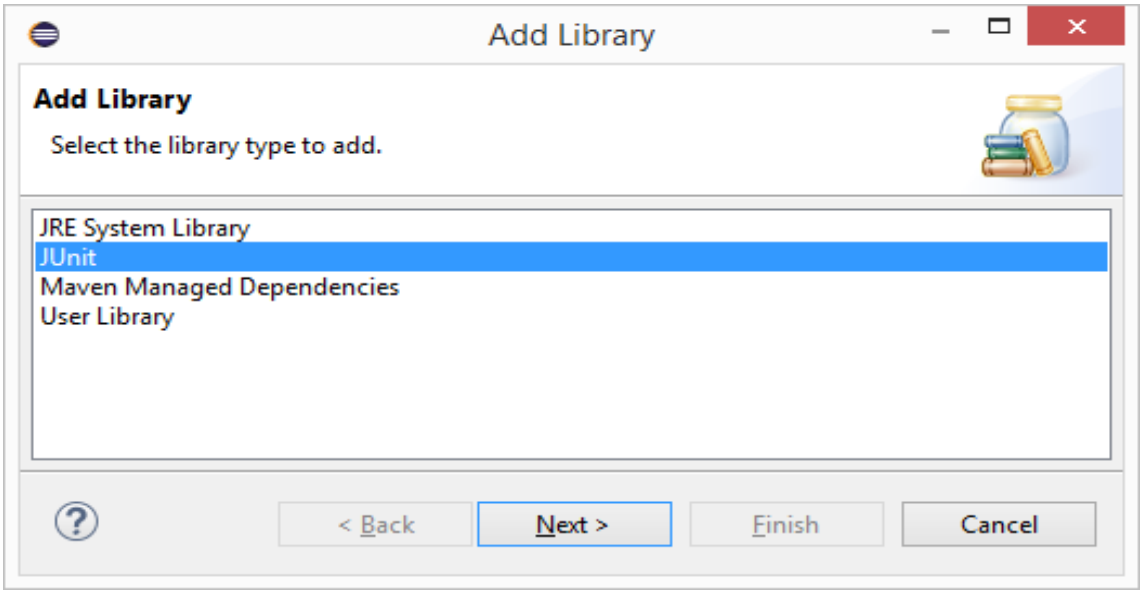
Com JUnit, o programador tem a possibilidade de usar esta ferramenta para criar um modelo padrão de testes, muitas vezes de forma automatizada.

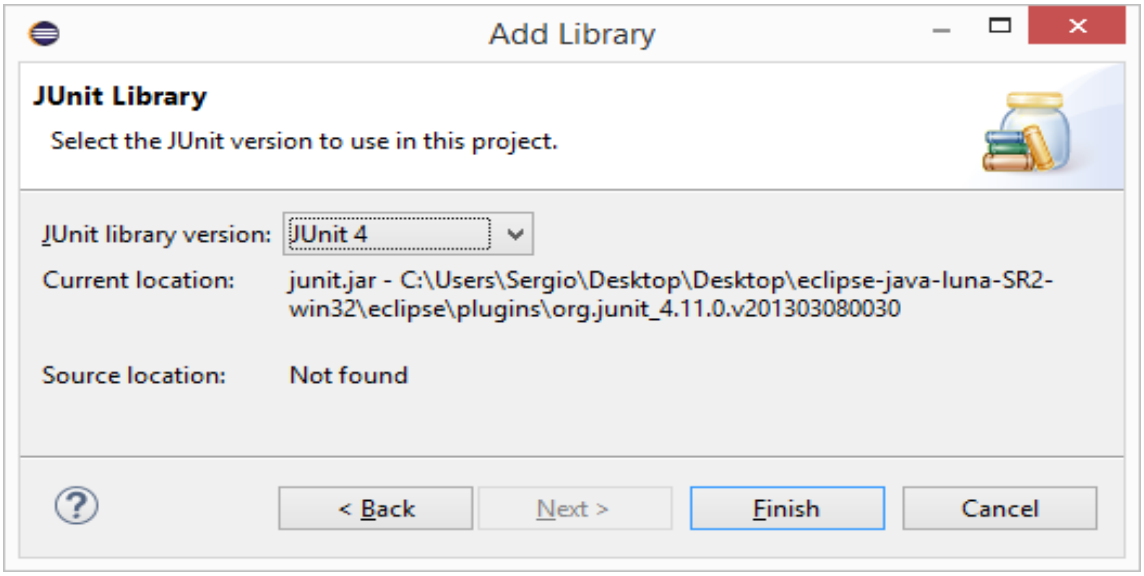
O teste de unidade testa o menor dos componentes de um sistema de maneira isolada. Cada uma dessas unidades define um conjunto de estímulos (chamada de métodos), e de dados de entrada e saída associados a cada estímulo. As entradas são parâmetros e as saídas são o valor de retorno, exceções ou o estado do objeto. Tipicamente um teste unitário executa um método individualmente e compara uma saída conhecida após o processamento da mesma

Adiciona o Junit ao projeto:




Properties for aula06_projeto03
type filter text
Resource
Builders
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project References
Run/Debug Settings
Task Repository
Task Tags
Validation
WikiText
Java Build Path
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
JRE System Library [JavaSE-1.8]
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
OK
Cancel


Add Library
Add Library
Select the library type to add.
JRE System Library
JUnit
Maven Managed Dependencies
User Library
< Back
Next >
Finish
Cancel


Add Library
JUnit Library
Select the JUnit version to use in this project.
JUnit library version:
JUnit 4
Current location:
junit.jar - C:\Users\Sergio\Desktop\Desktop\eclipse-java-luna-SR2-win32\eclipse\plugins\org.junit_4.11.0.v201303080030
Source location:
Not found
< Back
Next >
Finish
Cancel

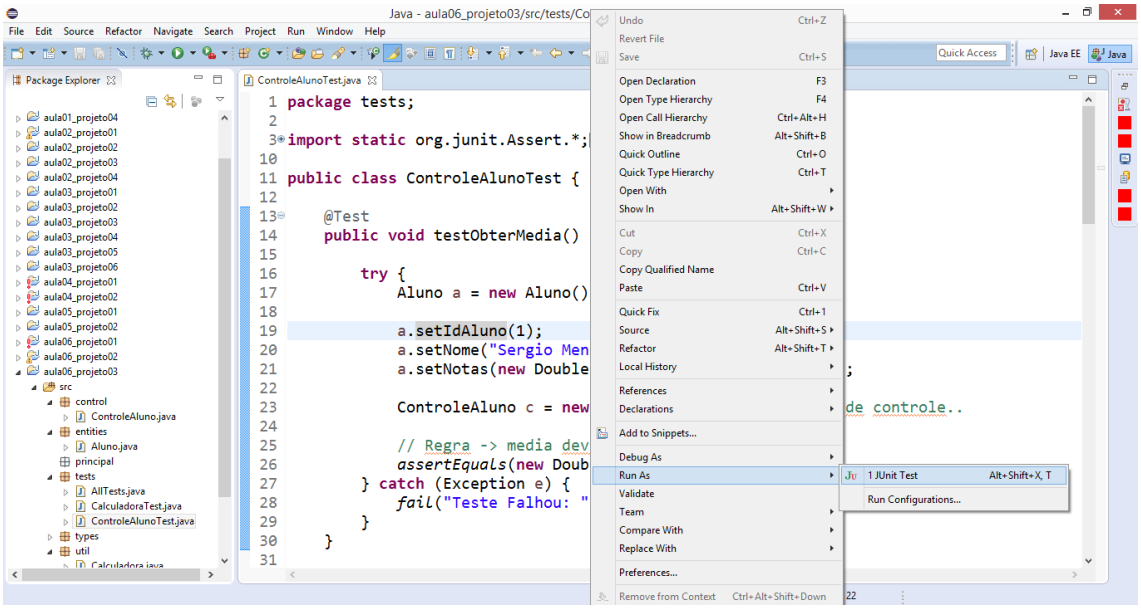
Criando uma Classe de testes para ControleAluno

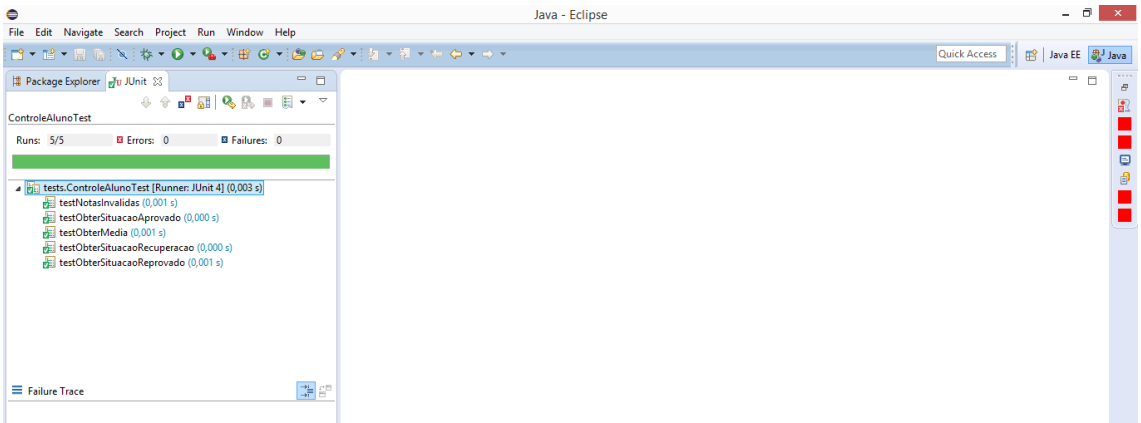
```
package tests;

import static org.junit.Assert.*;

import org.junit.Test;

import control.ControleAluno;
import entities.Aluno;
import types.SituacaoAluno;

public class ControleAlunoTest {

	@Test
	public void testObterMedia() {

		try {
			Aluno a = new Aluno(); // entidade..

			a.setIdAluno(1);
			a.setNome("Sergio Mendes");
			a.setNotas(new Double[] { 10.0, 8.0, 10.0, 8.0 });

			ControleAluno c = new ControleAluno(); // classe de controle..

			// Regra -> media deve ser: 9.0 (verificação)
			assertEquals(new Double(9.0), c.obterMedia(a));
		} catch (Exception e) {
			fail("Teste Falhou: " + e.getMessage());
		}
	}

	@Test
	public void testObterSituacaoAprovado() {

		Aluno a1 = new Aluno(1, "Ana", new Double[] { 7.0, 7.0, 7.0, 7.0 });
		Aluno a2 = new Aluno(2, "Rui", new Double[] { 10.0, 10.0, 10.0, 10.0 });

		ControleAluno c = new ControleAluno(); // classe de controle..

		try {
			// verificando se a situação dos alunos é 'Aprovado'
			assertEquals(SituacaoAluno.Aprovado, c.obterSituacao(a1));
			assertEquals(SituacaoAluno.Aprovado, c.obterSituacao(a2));

		} catch (Exception e) {
			fail("Teste Falhou: " + e.getMessage());
```

```
		}
	}

	@Test
	public void testObterSituacaoRecuperacao() {

		Aluno a1 = new Aluno(1, "Ana", new Double[] { 6.9, 6.9, 6.9, 6.9 });
		Aluno a2 = new Aluno(2, "Rui", new Double[] { 5.0, 5.0, 5.0, 5.0 });

		ControleAluno c = new ControleAluno(); // classe de controle..

		try {
			// verificando se a situação dos alunos é 'Recuperacao'
			assertEquals(SituacaoAluno.Recuperacao, c.obterSituacao(a1));
			assertEquals(SituacaoAluno.Recuperacao, c.obterSituacao(a2));

		} catch (Exception e) {
			fail("Teste Falhou: " + e.getMessage());
		}
	}

	@Test
	public void testObterSituacaoReprovado() {

		Aluno a1 = new Aluno(1, "Ana", new Double[] { 4.9, 4.9, 4.9, 4.9 });
		Aluno a2 = new Aluno(2, "Rui", new Double[] { 0.0, 0.0, 0.0, 0.0 });

		ControleAluno c = new ControleAluno(); // classe de controle..

		try {
			// verificando se a situação dos alunos é 'Reprovado'
			assertEquals(SituacaoAluno.Reprovado, c.obterSituacao(a1));
			assertEquals(SituacaoAluno.Reprovado, c.obterSituacao(a2));

		} catch (Exception e) {
			fail("Teste Falhou: " + e.getMessage());
		}
	}

	// método para verificar se é lançado uma exceção quando o aluno
	// possui notas fora do intervalo de 0 e 10
	@Test(expected = Exception.class)
	public void testNotasInvalidas() throws Exception {

		Aluno a1 = new Aluno(1, "Ana", new Double[] { -1.0, -1.0, -1.0, -1.0 });
		Aluno a2 = new Aluno(2, "Rui", new Double[] { 11.0, 11.0, 11.0, 11.0 });

		ControleAluno c = new ControleAluno(); // classe de controle..
```

```
            c.obterMedia(a1); //deve gerar uma exceção!
            c.obterMedia(a2); //deve gerar uma exceção!
        }
}
```

Executando:





Classe utilitária:

```
package util;

public class Calculadora {

      private Integer num1;
      private Integer num2;
```

```
	public Calculadora(Integer num1, Integer num2) {
		this.num1 = num1;
		this.num2 = num2;
	}

	public Integer somar(){
		return num1 + num2;
	}

	public Integer subtrair(){
		return num1 - num2;
	}

	public Integer multiplicar(){
		return num1 * num2;
	}

	public Integer dividir() throws Exception{
		if(num2 > 0){
			return num1 / num2;
		}
		else{
			throw new Exception("Erro. Divisão por zero.");
		}
	}

}
```

Testando a Classe Calculadora

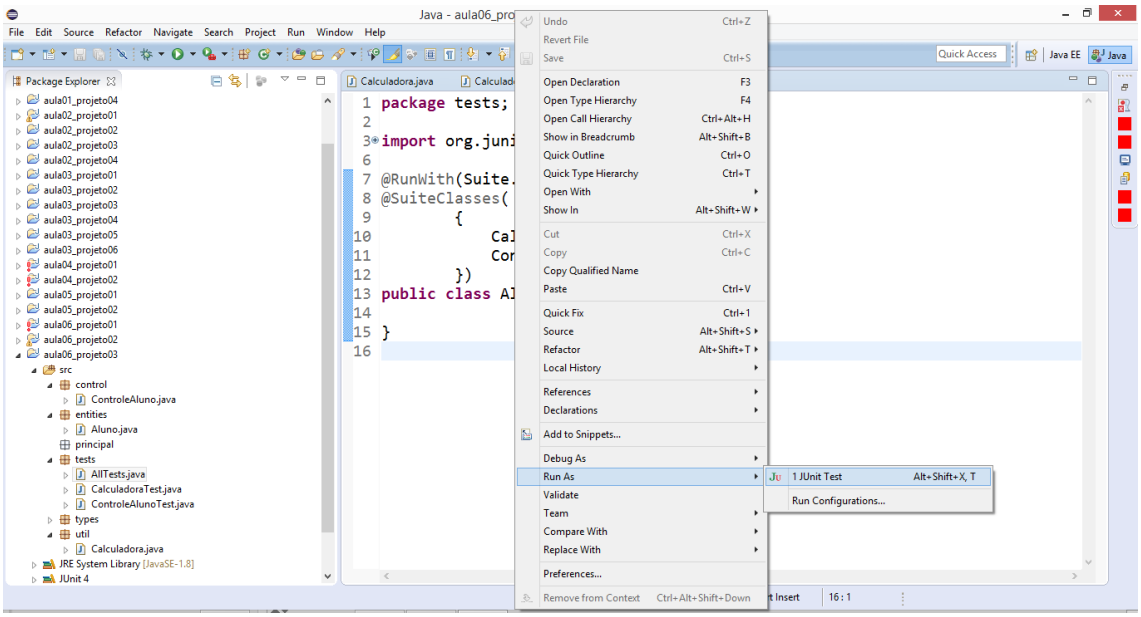
```
package tests;

import static org.junit.Assert.*;

import org.junit.Test;

import util.Calculadora;

public class CalculadoraTest {

	@Test
	public void testSomar() {
		Calculadora c = new Calculadora(10, 5);
		assertEquals(new Integer(15), c.somar());
	}
```

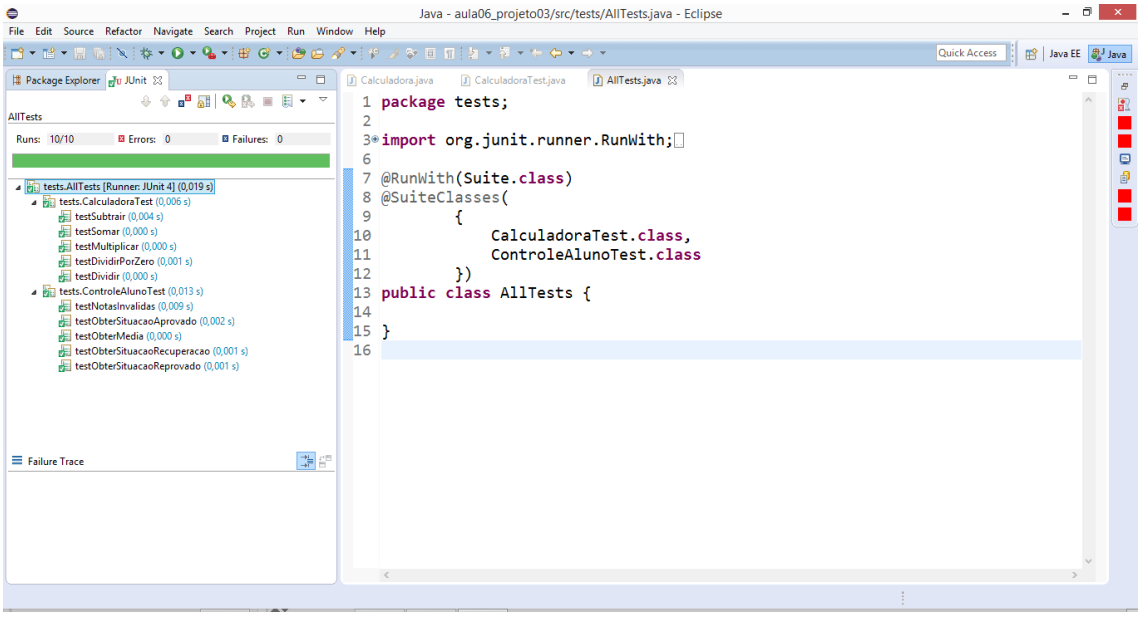
```
	@Test
	public void testSubtrair() {
		Calculadora c = new Calculadora(10, 5);
		assertEquals(new Integer(5), c.subtrair());
	}

	@Test
	public void testMultiplicar() {
		Calculadora c = new Calculadora(10, 5);
		assertEquals(new Integer(50), c.multiplicar());
	}

	@Test
	public void testDividir() {
		Calculadora c = new Calculadora(10, 5);

		try {
			assertEquals(new Integer(2), c.dividir());

		} catch (Exception e) {
			fail("Teste Falhou: " + e.getMessage());
		}
	}

	@Test(expected = Exception.class)
	public void testDividirPorZero() throws Exception{

		Calculadora c = new Calculadora(10, 0);
		c.dividir();
	}

}
```

Criando uma Suite de Testes para executar as demais classes e teste:

```
package tests;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses( {
		CalculadoraTest.class,   ControleAlunoTest.class
	})
public class AllTests {
}
```

Executando:


Java - aula06_pro
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
aula01_projeto04
aula02_projeto01
aula02_projeto02
aula02_projeto03
aula02_projeto04
aula03_projeto01
aula03_projeto02
aula03_projeto03
aula03_projeto04
aula03_projeto05
aula03_projeto06
aula04_projeto01
aula04_projeto02
aula05_projeto01
aula05_projeto02
aula06_projeto01
aula06_projeto02
aula06_projeto03
src
control
ControleAluno.java
entities
Aluno.java
principal
tests
AllTests.java
CalculadoraTest.java
ControleAlunoTest.java
types
util
Calculadora.java
JRE System Library [JavaSE-1.8]
JUnit 4
Calculadora.java
1 package tests;
3 import org.juni
7 @RunWith(Suite.
8 @SuiteClasses(
9 {
10 Cal
11 Cor
12 })
13 public class Al
15 }
Undo Ctrl+Z
Revert File
Save Ctrl+S
Open Declaration F3
Open Type Hierarchy F4
Open Call Hierarchy Ctrl+Alt+H
Show in Breadcrumb Alt+Shift+B
Quick Outline Ctrl+O
Quick Type Hierarchy Ctrl+T
Open With
Show In Alt+Shift+W
Cut Ctrl+X
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Quick Fix Ctrl+1
Source Alt+Shift+S
Refactor Alt+Shift+T
Local History
References
Declarations
Add to Snippets...
Debug As
Run As
Validate
Team
Compare With
Replace With
Preferences...
Remove from Context Ctrl+Alt+Shift+Down
1 JUnit Test Alt+Shift+X, T
Run Configurations...
Quick Access
Java EE
Java


Java - aula06_projeto03/src/tests/AllTests.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
AllTests
Runs: 10/10
Errors: 0
Failures: 0
tests.AllTests [Runner: JUnit 4] (0,019 s)
tests.CalculadoraTest (0,006 s)
testSubtrair (0,004 s)
testSomar (0,000 s)
testMultiplicar (0,000 s)
testDividirPorZero (0,001 s)
testDividir (0,000 s)
tests.ControleAlunoTest (0,013 s)
testNotasInvalidas (0,009 s)
testObterSituacaoAprovado (0,002 s)
testObterMedia (0,000 s)
testObterSituacaoRecuperacao (0,001 s)
testObterSituacaoReprovado (0,001 s)
Failure Trace
Calculadora.java
CalculadoraTest.java
AllTests.java
1 package tests;
3 import org.junit.runner.RunWith;
7 @RunWith(Suite.class)
8 @SuiteClasses(
9 {
10 CalculadoraTest.class,
11 ControleAlunoTest.class
12 })
13 public class AllTests {
15 }
Quick Access
Java EE
Java