




**BEM VINDOS!**  
**VENTURUS<sup>4</sup>TECH**



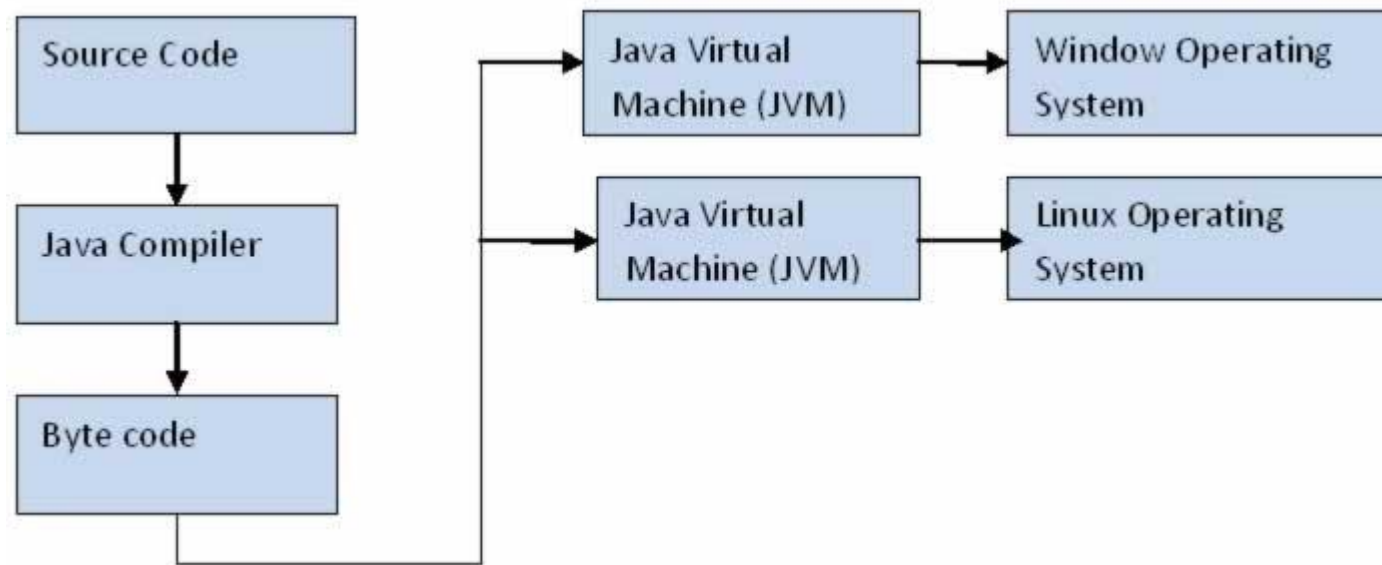
# Módulo I

## Introdução ao Java

**VENTURUS4TECH**

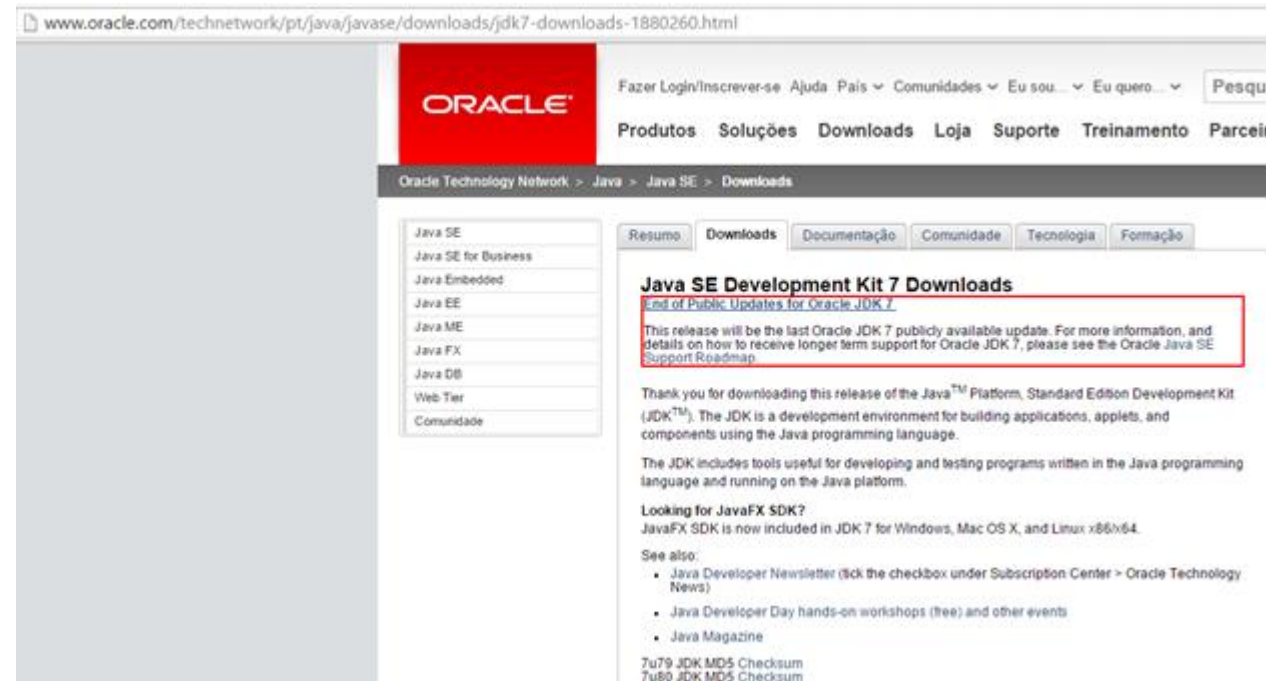
Com Maurício Schiezero  
[mauricio.schiezero@venturus.org.br](mailto:mauricio.schiezero@venturus.org.br)

# Arquitetura



# Java SDK

<http://www.oracle.com/technetwork/pt/java/javase/downloads/jdk7-downloads-1880260.html>



The screenshot shows the Oracle Technology Network page for Java SE Development Kit 7 Downloads. The page is in Portuguese. The Oracle logo is in the top left. The top navigation bar includes links for 'Fazer Login/Inscrever-se', 'Ajuda', 'País', 'Comunidades', 'Eu sou...', 'Eu quero...', and a search bar. Below this is a secondary navigation bar with 'Produtos', 'Soluções', 'Downloads', 'Loja', 'Suporte', 'Treinamento', and 'Parceiros'. The breadcrumb trail reads 'Oracle Technology Network > Java > Java SE > Downloads'. On the left, a sidebar lists various Java products: 'Java SE', 'Java SE for Business', 'Java Embedded', 'Java EE', 'Java ME', 'Java FX', 'Java DB', 'Web Tier', and 'Comunidade'. The main content area has tabs for 'Resumo', 'Downloads', 'Documentação', 'Comunidade', 'Tecnologia', and 'Formação'. The 'Downloads' tab is selected, showing the title 'Java SE Development Kit 7 Downloads'. A red box highlights the text: 'End of Public Updates for Oracle JDK 7. This release will be the last Oracle JDK 7 publicly available update. For more information, and details on how to receive longer term support for Oracle JDK 7, please see the Oracle Java SE Support Roadmap.' Below this, there is a thank you message, information about the JDK, and a section for 'Looking for JavaFX SDK?'. At the bottom, there are links to 'See also:' including 'Java Developer Newsletter', 'Java Developer Day hands-on workshops', and 'Java Magazine'. The page also lists '7u79 JDK MD5 Checksum' and '7u80 JDK MD5 Checksum'.

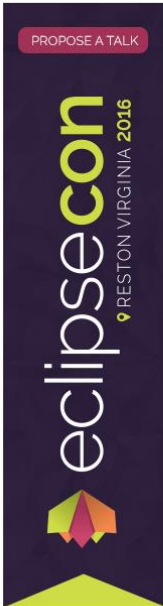
# Arquitetura

<https://eclipse.org/>


[HOME](#) / [DOWNLOADS](#)

[» Packages](#) [Developer Builds](#)

[PROPOSE A TALK](#)



Eclipse Mars.1 (4.5.1) Release for Windows



## Try the Eclipse Installer

NEW

The easiest way to install and update your Eclipse Development Environment.


[FIND OUT MORE](#)

Mac OS X  
64 bit

Windows  
32 bit | 64 bit

Linux  
32 bit | 64 bit

### ...or download an Eclipse Package




#### Eclipse IDE for Java EE Developers

275 MB | 1,300,884 DOWNLOADS

Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn...


Windows  
32 bit | 64 bit



#### JRebel for Eclipse IDE

See Java Code Changes Instantly. Save Time. Reduce Stress. Finish Projects Faster!

Promoted Download




#### Eclipse IDE for Java Developers

166 MB | 632,632 DOWNLOADS

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven integration and WindowBuilder...

Windows  
32 bit | 64 bit



#### Gradle Integration from Buildship

Eclipse plug-ins that provide support for building software using Gradle.

[INSTALL NOW](#)

#### RELATED LINKS

- Compare & Combine Packages
- New and Noteworthy
- Install Guide
- Documentation
- Updating Eclipse
- Forums

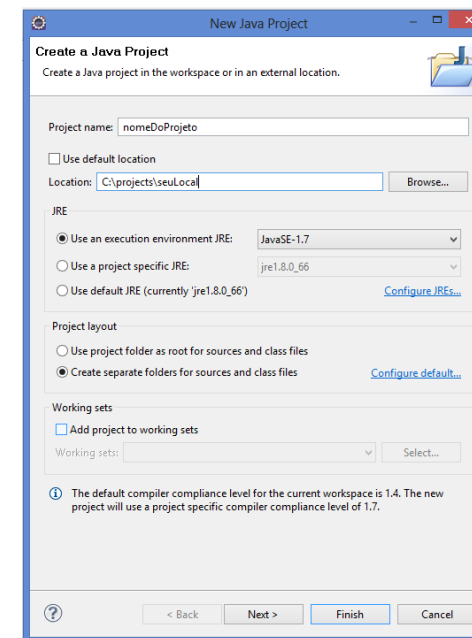
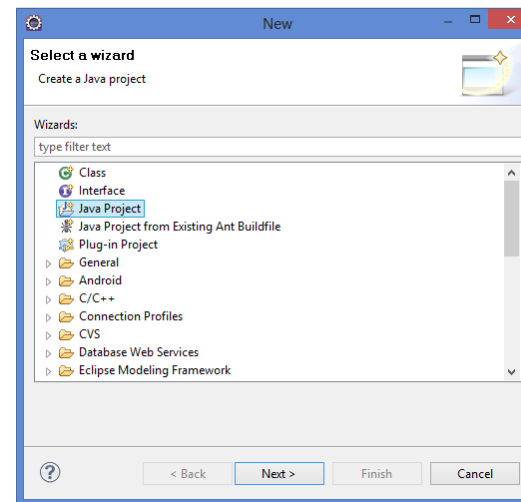
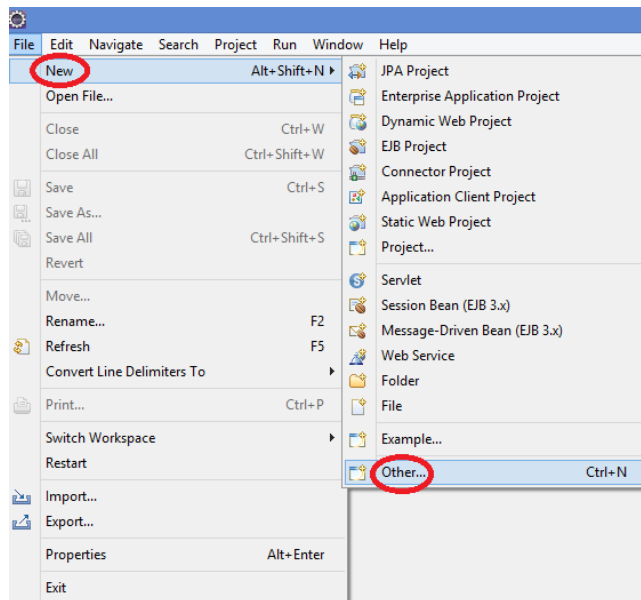
#### MORE DOWNLOADS

- Other builds
- Eclipse Mars (4.5)
- Eclipse Luna (4.4)
- Eclipse Kepler (4.3)
- Eclipse Juno (4.2)
- Older Versions

VENTURUS<sup>4</sup>TECH  
JANEIRO/16

# Criando um projeto

- Criar uma pasta para o projeto
- Criar o diretório src
- Criar o projeto no eclipse conforme a figura
- Apontar o Location para a pasta do projeto



# Hello World

```
package br.org.ventururs.training;  
  
public class HelloWorld {  
  
    public static void main(String[] args) {  
  
        System.out.println("Hello World");  
  
    }  
}
```

# Tipos Primitivos

```
byte myByte= 126;      // -128 a 127  8 bits  
short myShort = 13;    // -32768 a 32767 16 bits  
int myInt = 13;         // -2^31 a 2^31  32 bits  
long myLong = 45;       // -2^63 a 2^63  64 bits  
float myFloat = 5.6f;   // 32 bits  
double myDouble= 4.8;   // 64 bits
```

```
boolean b = true;  
char c = 'a';      // 16 bits
```

```
int sum = 4 + 5;  
int sub = 4 - 5;  
int mult = 4 * 5;  
int div = 13 / 5;  
int mod = 13 % 5;
```

```
System.out.println(sum);  
System.out.println(sub);  
System.out.println(mult);  
System.out.println(div);  
System.out.println(mod);
```



# Strings

```
// Create a string with a constructor
String s1 = new String("Who let the dogs out?");
// Just using "" creates a string, so no need to write it the previous way.
String s2 = "Who who who who!";
// Java defined the operator + on strings to concatenate:
String s3 = s1 + s2;

System.out.println(s3);
```

# If / else

```
boolean condition = false;
if (condition) {
    System.out.println("True");
} else {
    System.out.println("False");
}
```

# Arrays / for

```
int[] array = new int[5];  
int[] array2 = {1,2,3,4,5};  
  
System.out.println("for");  
for (int i = 0; i < array2.length; i++) {  
    System.out.println(array2[i]);  
}
```

# do / while

- Execute o código
- Troque stopCondition = 0
- Por que o resultado dos dois laços é diferente ?

```
int stopCondition = 5;

int j = 0;
System.out.println("While");
while (j++ < stopCondition) {
    System.out.print("+");
}
System.out.println("");
int i = 0;
System.out.println("Do");
do {
    System.out.print("+");
} while (++i < stopCondition);
```

# Boolean

```
int a = 4;
int b = 5;
boolean result;
result = a < b; // true
System.out.println(result);
result = a > b; // false
System.out.println(result);
result = a <= 4 ;// a smaller or equal to 4 - true
System.out.println(result);
result = b >= 6 ;// b bigger or equal to 6 - false
System.out.println(result);
result = a == b ;// a equal to b - false
System.out.println(result);
result = a != b ;// a is not equal to b - true
System.out.println(result);
result = a > b || a < b ;// Logical or - true
System.out.println(result);
result = 3 < a && a < 6 ;// Logical and - true
System.out.println(result);
result = !result ;// Logical not - false
System.out.println(result);
```

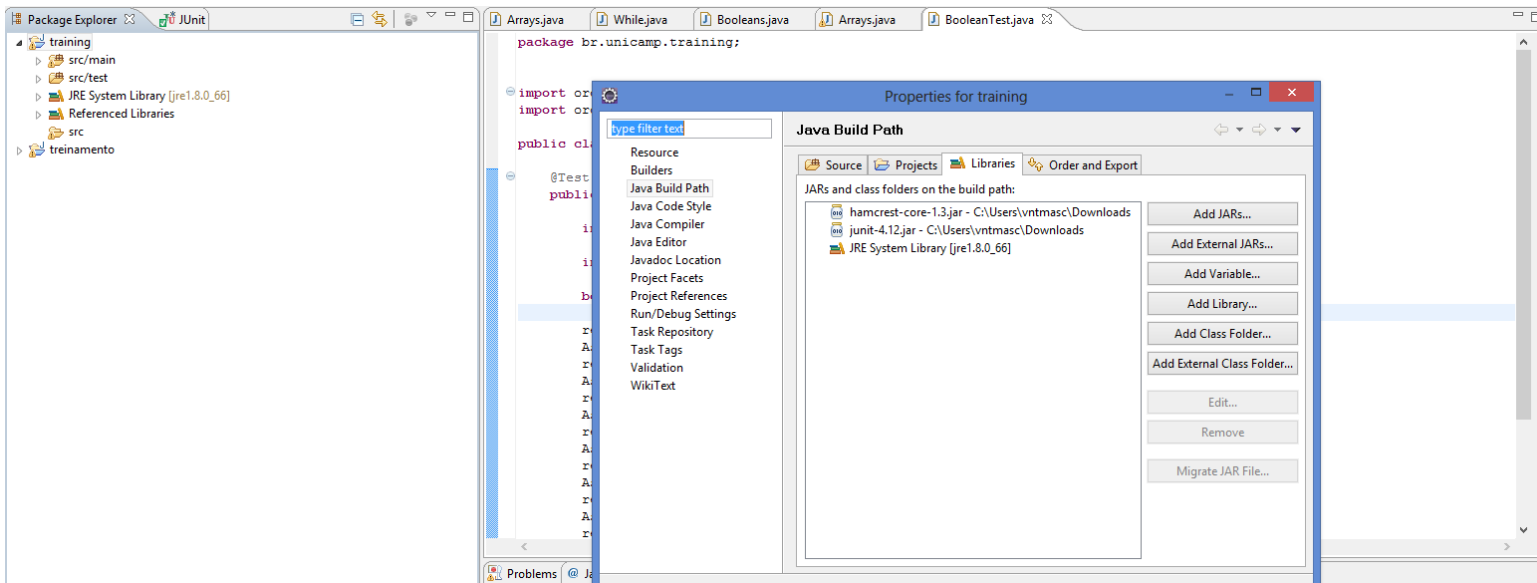
# Junit

- <http://junit.org/>
- JUnit is a simple framework to write repeatable tests

```
import org.junit.Assert;  
import org.junit.Test;
```

```
@Test  
public void testBoolean() {  
  
    Assert.assertTrue(  

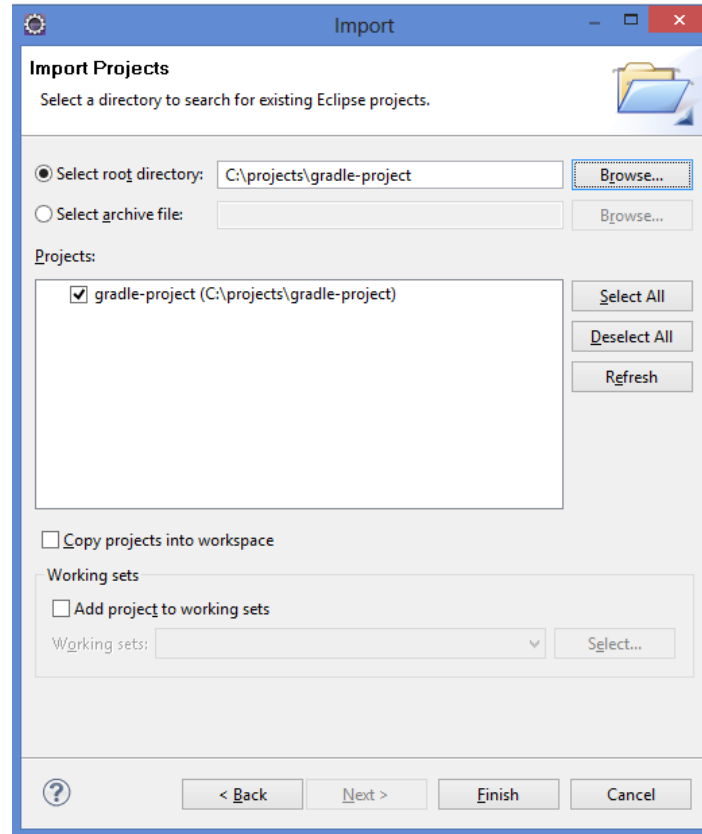
```



# Gradle

- <http://gradle.org/gradle-download/>
- Criar variável GRADLE\_HOME
- Colocar no path \$GRADLE\_HOME/bin
- Criar diretório do projeto gradle
- Entrar no diretório criado
- `gradle init --type java-library`
- Abrir arquivo build.gradle
- Adicionar dependência do plugin para gerar projeto do eclipse
- `apply plugin: 'eclipse'` (logo abaixo do `apply plugin: 'java'`)
- Entrar no diretório novamente e digitar `gradle eclipse`
- No eclipse entrar em file -> Import -> General -> Existing Projects into Workspace
  - Seleciona o diretório do projeto e aparecerá a opção do projeto gerado

# Gradle





# Orientação a objetos

## Classes

```
public class Class {  
  
}
```

## Atributos

```
public class Class {  
    private String attribute;  
}
```

## Objetos

```
Class c = new Class();
```

## Métodos

```
public class Class {  
    private int sum;  
  
    public void method1() {  
        System.out.println("Method 1");  
    }  
  
    public int addValue(int value) {  
        sum += value;  
        return sum;  
    }  
}
```

# Orientação a objetos

## Modificadores de acesso para classes

- Sem modificador (**package**) acesso somente dentro do mesmo pacote (diretório) — `class Coordinate {`
- **public** — `public class Coordinate {`

## Modificadores de acesso para atributos e métodos

- Sem modificador (**package**)
- **private** acesso somente dentro da própria classe
- **public** acesso irrestrito de qualquer classe — `Coordinate c = new Coordinate();`
- **protected** acesso somente nas classes filhas (herança) `int a = c.x;`

# Atributos private e public

- Modifique a classe Coordinate para que o teste CoordinateTestCase passe
  - Existem problemas relacionados a acessibilidade de atributos e da classe

```
package br.org.ventururs;

import org.junit.Assert;

import br.org.ventururs.training.Coordinate;

public class CoordinateTestCase {
    public void testCoordinateCreation() {

        Coordinate c = new Coordinate();

        c.x = 4;

        Assert.assertEquals(4,c.x);
        Assert.assertEquals(7,c.y);
    }
}
```

```
package br.org.ventururs.training;

class Coordinate {

    int x;

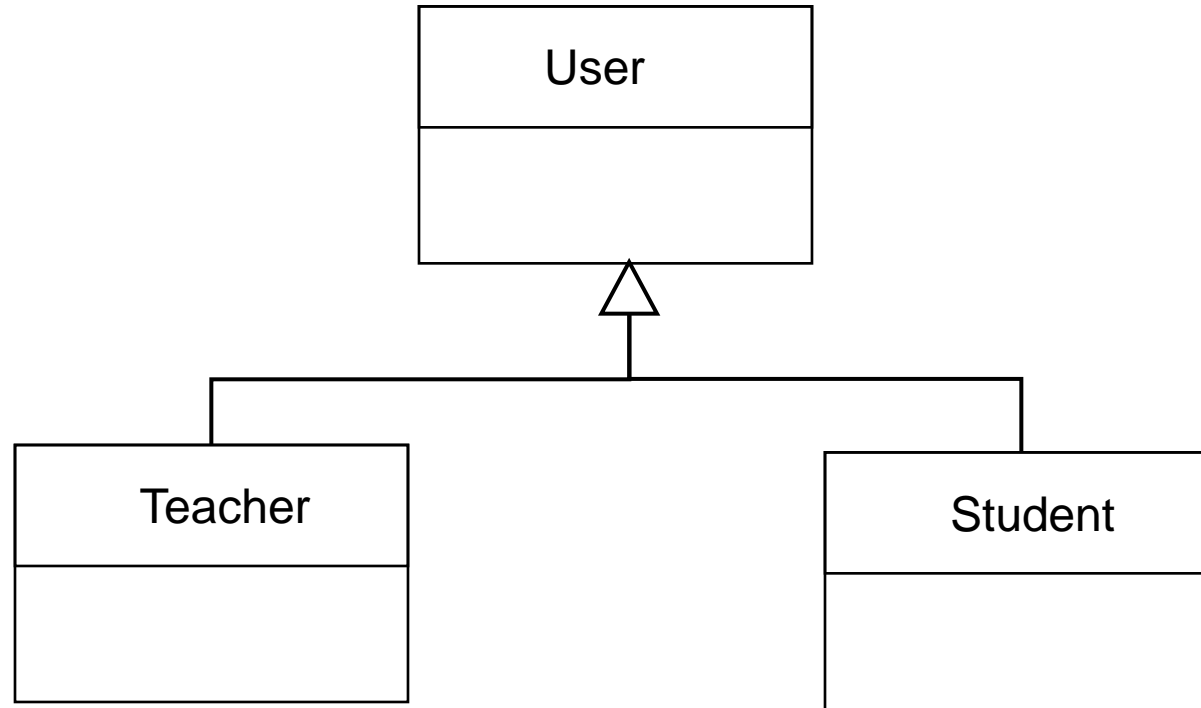
    private int y;

}
```

# Atributos private e public

- Agora volte os atributos da classe Coordinate para privado
  - Crie métodos para acessar os valores de x e y
  - Crie métodos para modificar os valores de x e y e / ou crie um Construtor passando x e y como parâmetros para atualizar os valores dos atributos

# Herança



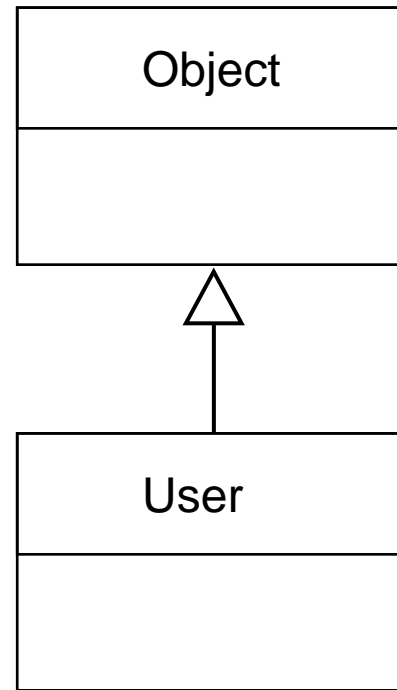
# Herança

```
public class User {  
    protected String userName;  
    protected String password;  
    protected String firstName;  
    protected String lastName;  
    protected int age;  
    public User(String userName, String password, String firstName,  
                String lastName, int age) {  
        this.userName = userName;  
        this.password = password;  
        this.firstName = firstName;  
        this.lastName = lastName;  
        this.age = age;  
    }  
    @Override  
    public String toString() {  
        return "User [userName=" + userName + ", password=" + password  
            + ", firstName=" + firstName + ", lastName=" + lastName  
            + ", age=" + age + "];"  
    }  
}
```

```
package br.org.ventururs.training;  
  
public class Student extends User {  
    private String course;  
}
```

```
package br.org.ventururs.training;  
  
public class Teacher extends User {  
    private String department;  
}
```

# Herança



# Herança

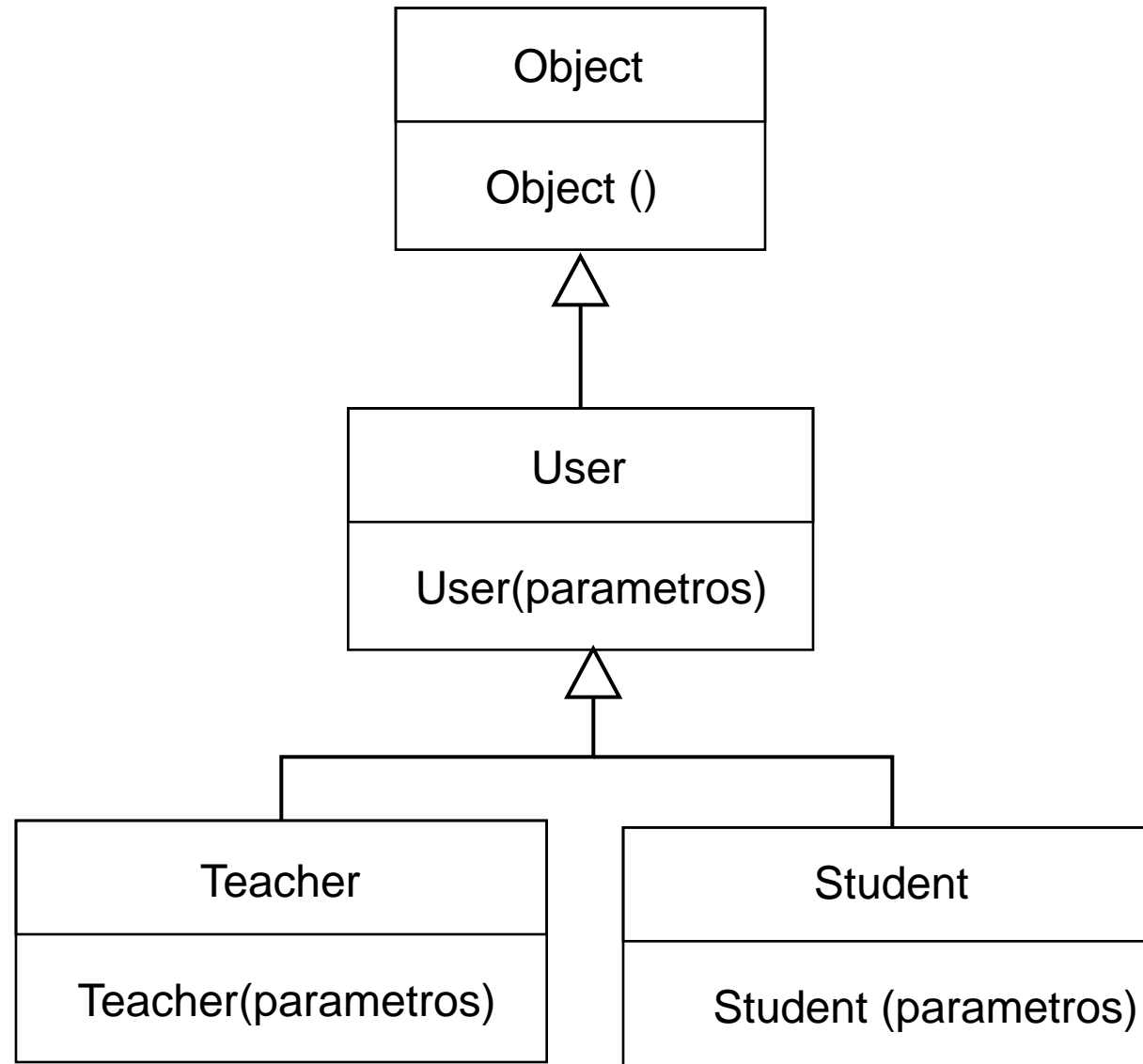
Modifier and Type	Method and Description
protected <b>Object</b>	<b>clone()</b> Creates and returns a copy of this object.
boolean	<b>equals(Object obj)</b> Indicates whether some other object is "equal to" this one.
protected void	<b>finalize()</b> Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
<b>Class&lt;?&gt;</b>	<b>getClass()</b> Returns the runtime class of this <b>Object</b> .
int	<b>hashCode()</b> Returns a hash code value for the object.
void	<b>notify()</b> Wakes up a single thread that is waiting on this object's monitor.
void	<b>notifyAll()</b> Wakes up all threads that are waiting on this object's monitor.
<b>String</b>	<b>toString()</b> Returns a string representation of the object.
void	<b>wait()</b> Causes the current thread to wait until another thread invokes the <b>notify()</b> method or the <b>notifyAll()</b> method for this object.
void	<b>wait(long timeout)</b> Causes the current thread to wait until either another thread invokes the <b>notify()</b> method or the <b>notifyAll()</b> method for this object, or a specified amount of time has elapsed.
void	<b>wait(long timeout, int nanos)</b> Causes the current thread to wait until another thread invokes the <b>notify()</b> method or the <b>notifyAll()</b> method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed.



# Herança

- Criar construtores para as classes Student e Teacher (corrigirá os erros de compilação de UsersTestCase)
- (Clicar botão direito -> source-> Generate construtor using fields)

# Herança - Construtores



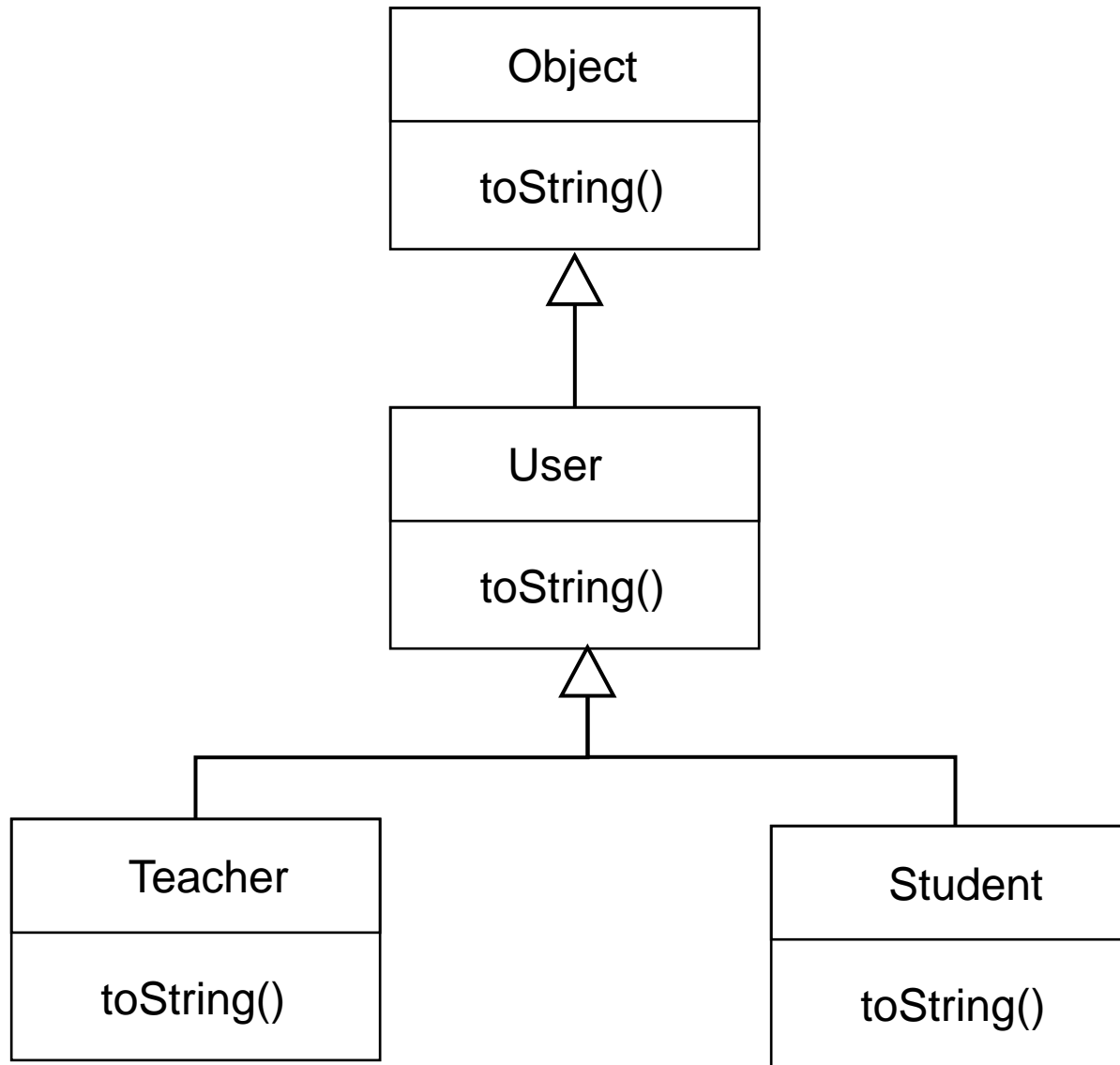
# Herança - Construtores

- Rodar o testes UsersTestCase
- Todos os testes passaram ?
- O que aconteceu ?

# Herança - Override - toString

- Método do classe Object
- Quando executamos **System.out.println** e passamos uma classe como parâmetro é o método toString que é executado para mostrarmos o seu valor
- Se você criar uma classe e não implementar o método toString, toda vez executarmos seu método toString será executado o método toString da classe pai
- Se a classe pai for Object o resultado será o endereço de memória da classe

# Herança - Override - toString



# Herança - Override - toString

- Os testes que falharam são por problemas de criação do método toString
- Criar o método toString para as classes que estão com problemas e re-executar os testes
- Dica clique com o botão direito -> source -> Generate toString()

```
@Test
public void testUserToString() {
    User user = new User("username", "password", "firstName", "lastName", 15);

    String expectedToString =
        "User [userName=username, password=password, firstName=firstName, "
        + "lastName=lastName, age=15]";

    String actualToString = user.toString();

    Assert.assertEquals(expectedToString, actualToString);
}

@Test
public void testTeacherToString() {
    Teacher teacher = new Teacher("username", "password", "firstName", "lastName", 15, "department");

    String expectedToString =
        "Teacher [department=department, userName=username, password=password, firstName=firstName, "
        + "lastName=lastName, age=15]";

    String actualToString = teacher.toString();

    Assert.assertEquals(expectedToString, actualToString);
}
```

# Herança -Modificadores

- Troque um atributo da classe User para private
- O que acontece ?

# Polimorfismo

- Parâmetro pode ser uma instância de:
  - User
  - Teacher
  - Student
- Apesar do parâmetro ser User o método toString chamado será o da instância que foi passado por parâmetro e não da classe User

```
package br.org.ventururs.training;  
  
public class Polimorfism {  
  
    public String printUserToString(User user) {  
        return user.toString();  
    }  
  
}
```



# Sobrecarga de métodos

```
package br.org.ventururs.training;  
  
public class Polimorfism {  
  
    public String printUserToString(User user) {  
        return user.toString();  
    }  
  
    public String printUserToString(User user, String otherParam) {  
        return user.toString()+otherParam;  
    }  
  
    public String printUserToString(Object o) {  
        return o.toString();  
    }  
  
}
```

# Métodos estáticos

- Colocar o modificador static na frente do método printUserToString da classe Polimorfism
- Altere a classe de testes com a chamada estática

```
Polimorfism.printUserToString(user);
```

- Rode o teste em modo debug para ver quais métodos de quais classes são chamados

```
package br.org.ventururs.training;
```

```
public class Polimorfism {
```

```
    public String printUserToString(User user) {  
        return user.toString();  
    }
```

```
}
```

```
Polimorfism p = new Polimorfism();  
p.printUserToString(user);
```

# Métodos estáticos

- Altere a classe de testes com a chamada estática que só usem o método estático ?
- Como eu não permito que as pessoas criem classes filhas de Polimorfism ?

# Métodos estáticos

- Altere a classe de testes com a chamada estática que só usem o método estático ?
  - Construtor private
- Como eu não permito que as pessoas criem classes filhas de Polimorfism ?
  - Classe final

# Métodos estáticos

- Cria uma nova classe Test com o método main
  - Crie uma variável que receba uma instância de Polimorfism (new Polimorfism)
  - Crie um construtor vazio (botão direito -> source -> Generate constructor from super classe)
  - Modifique o construtor para private
  - Qual é o resultado ?
- 
- Volte o construtor para public
  - Crie uma nova classe filha de Polimorfism
  - Coloque o modificador final na classe
  - O que acontece ?
- 
- Volte o construtor para public
  - Modifique o construtor para private
  - Qual é o resultado ?

# Classes abstratas

- Transformar classe User em abstrata `public abstract class User {`
- Remover os testes unitários que instanciam a classe User
- Criar um método abstrato `String printType()` `public abstract String printType();`
- Que imprime o nome da classe
- Fazer o teste unitário para classe Student `String printType()`
- Esse método poderia ser feito na classe abstrata ?
  - Mova o método `String printType()` e retire das classes filhas

```
public abstract class AbstractClass {  
    private int attribute;  
    public abstract void abstractMethod();  
    public void increment() {  
        attribute++;  
    }  
}  
  
public class NewClass extends AbstractClass {  
    @Override  
    public void abstractMethod() {  
    }  
}
```

# Interfaces

- Parecida com classe abstrata, mas não tem implementações
  - Somente assinatura de métodos

```
package br.org.ventururs.training;  
  
public interface Expression {  
  
    public Object getValue();  
  
}
```

```
package br.org.ventururs.training;  
  
public class SumExpression implements Expression {  
  
    private Double operand1;  
  
    private Double operand2;  
  
    public SumExpression(Double operand1, Double operand2) {  
        this.operand1 = operand1;  
        this.operand2 = operand2;  
    }  
  
    @Override  
    public Object getValue() {  
  
        return operand1 + operand2;  
    }  
}
```

# Generics

- Alterar Expression para usar Generics
- Criar expressão GreatherThan que retorna um Boolean
- Criar um método para teste unitário igual foi feito para o SumExpression

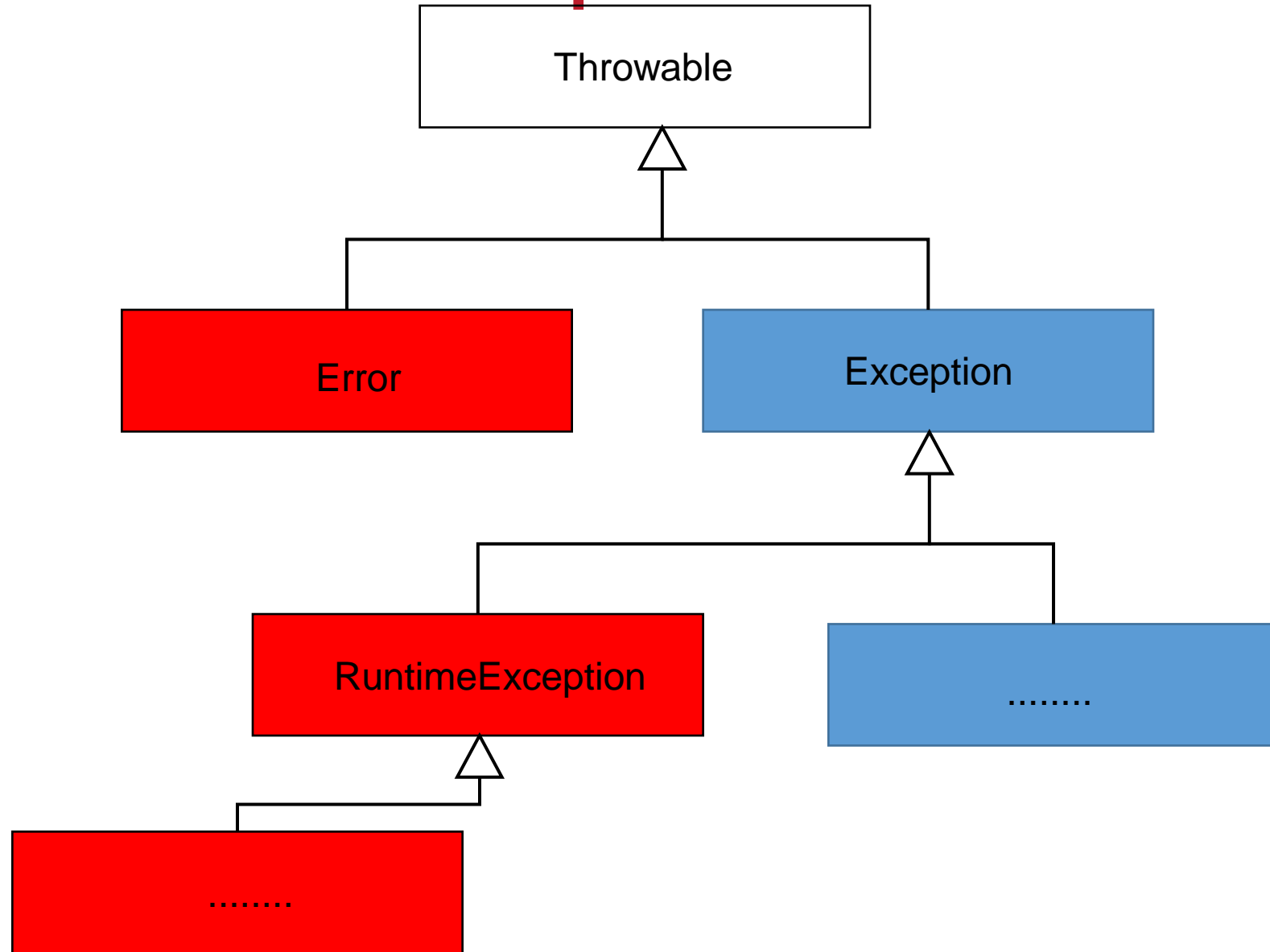
```
package br.org.ventururs.training;  
  
public interface Expression {  
    public Object getValue();  
}
```

```
package br.org.ventururs.training;  
  
public interface Expression<T> {  
    public T getValue();  
}
```

```
package br.org.ventururs.training;  
  
public class SumExpression implements Expression<Double> {  
    private Double operand1;  
    private Double operand2;  
  
    public SumExpression(Double operand1, Double operand2) {  
        this.operand1 = operand1;  
        this.operand2 = operand2;  
    }  
  
    @Override  
    public Double getValue() {  
        return operand1 + operand2;  
    }  
}
```



# Exceptions



# Exceptions

<https://docs.oracle.com/javase/7/docs/api/>

java.lang

## Class Exception

java.lang.Object

java.lang.Throwable

java.lang.Exception

### All Implemented Interfaces:

Serializable

### Direct Known Subclasses:

AcNotFoundException, ActivationException, AlreadyBoundException, ApplicationException, AWTException, BackingStoreException, BadAttributeValueExpException, BadBinaryOpValueExpException, BadLocationException, BadStringOperationException, BrokenBarrierException, CertificateException, CloneNotSupportedException, DataFormatException, DatatypeConfigurationException, DestroyFailedException, ExecutionException, ExpandVetoException, FontFormatException, GeneralSecurityException, GSSEException, IllegalClassFormatException, InterruptedException, IntrospectionException, InvalidApplicationException, InvalidMidiDataException, InvalidPreferencesFormatException, InvalidTargetObjectTypeException, IOException, JAXBException, JMEException, KeySelectorException, LastOwnerException, LineUnavailableException, MarshalException, MidiUnavailableException, MimeTypeParseException, MimeTypeParseException, NamingException, NoninvertibleTransformException, NotBoundException, NotOwnerException, ParseException, ParserConfigurationException, PrinterException, PrintException, PrivilegedActionException, PropertyVetoException, ReflectiveOperationException, RefreshFailedException, RemarshalException, RuntimeException, SAXException, ScriptException, ServerNotActiveException, SOAPException, SQLException, TimeoutException, TooManyListenersException, TransformerException, TransformException, UnmodifiableClassException, UnsupportedAudioFileException, UnsupportedCallbackException, UnsupportedFlavorException, UnsupportedLookAndFeelException, URISyntaxException, URIReferenceException, UserException, XAException, XMLParseException, XMLSignatureException, XMLStreamException, XPathException

# Exceptions

<https://docs.oracle.com/javase/7/docs/api/>

java.lang

## Class RuntimeException

java.lang.Object

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

### All Implemented Interfaces:

Serializable

### Direct Known Subclasses:

AnnotationTypeMismatchException, ArithmeticException, ArrayStoreException, BufferOverflowException, BufferUnderflowException, CannotRedoException, CannotUndoException, ClassCastException, CMMException, ConcurrentModificationException, DataBindingException, DOMException, EmptyStackException, EnumConstantNotPresentException, EventException, FileSystemAlreadyExistsException, FileSystemNotFoundException, IllegalArgumentException, IllegalMonitorStateException, IllegalPathStateException, IllegalStateException, IllformedLocaleException, ImagingOpException, IncompleteAnnotationException, IndexOutOfBoundsException, JMRuntimeException, LSEException, MalformedParameterizedTypeException, MirroredTypesException, MissingResourceException, NegativeArraySizeException, NoSuchElementException, NoSuchMechanismException, NullPointerException, ProfileDataException, ProviderException, ProviderNotFoundException, RasterFormatException, RejectedExecutionException, SecurityException, SystemException, TypeConstraintException, TypeNotPresentException, UndeclaredThrowableException, UnknownEntityException, UnmodifiableSetException, UnsupportedOperationException, WebServiceException, WrongMethodTypeException

# RuntimeException

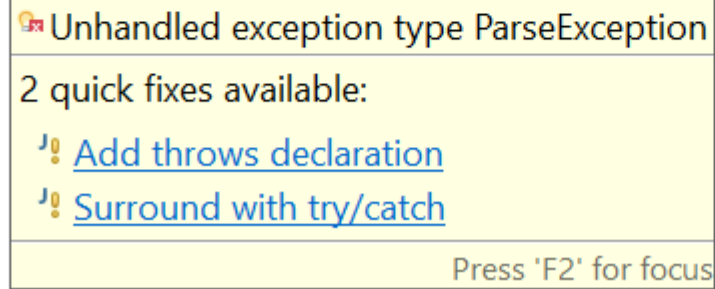
```
String[] array = {"one", "two", "three"};
```

```
System.out.println(array[3]);
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3  
    at br.org.ventururs.training.Exceptions.main(Exceptions.java:9)
```



# Exception

```
package br.org.ventururs.training;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class DateUtil {  
    private final static SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");  
  
    public static Date formatDate(String date) {  
        return sdf.parse(date);  
    }  
}
```



Unhandled exception type ParseException

2 quick fixes available:

-  [Add throws declaration](#)
-  [Surround with try/catch](#)

Press 'F2' for focus

# Exception

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateUtil {

    private final static SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

    public static Date formatDate(String date) throws ParseException {

        return sdf.parse(date);
    }
}
```

# Exception

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateUtil {

    private final static SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

    public static Date format(String date) {
        Date ret = null;
        try {
            ret = sdf.parse(date);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        return ret;
    }
}
```

# Criando sua Exception

```
public class NewException extends Exception {
```

```
public class NewRuntimeException extends RuntimeException {
```



# Exercício Exception

```
public class Account {  
  
    private String id;  
  
    private String bankCode;  
  
    private Double amount;  
  
    private User user;  
  
    public Double deposit (double value) {  
        amount += value ;  
        return amount;  
    }  
  
    public Double withdraw(double value) {  
        amount -= value ;  
        return amount;  
    }  
}
```

# Exercício Exception

```
public class AccountManager {  
    public static void transfer(double value, Account origin, Account target) {  
        if(! origin.getId().equals(target.getId())) {  
            origin.withdraw(value);  
            target.deposit(value);  
        }  
    }  
}
```

# Exercício Exception

- Na classe AccountTestCase o primeiro teste está correto, entenda o código do teste para verificar se a transferência foi bem feita
- O segundo método o saldo da primeira conta (origem) vai ficar negativa. Crie uma Exception (InvalidAmountException) para ser lançada quando não houver saldo na conta.
- Modifique a classe Account para lançar essa exception
- Faça o teste unitário passar

# Equals e hashCode

- Métodos utilizados para testar a igualdade dos objetos
- Não implementar / conhecer o comportamento desses métodos dificulta o uso de Collections (ex: HashMap e Set)

```
@Override  
public int hashCode() {
```

```
@Override  
public boolean equals(Object obj) {
```

# Equals e hashCode

- Exemplo. Quais atributos vc usaria no método equals ?

```
public class Person {  
    private String firtsName;  
    private String lastName;  
    private String rg;  
    private String cpf;
```

# Equals e hashCode

- Exemplo. Quais atributos vc usaria no método equals ?

```
@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((cpf == null) ? 0 : cpf.hashCode());
    return result;
}
```

```
@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    Person other = (Person) obj;
    if (cpf == null) {
        if (other.cpf != null)
            return false;
    } else if (!cpf.equals(other.cpf))
        return false;
    return true;
}
```

# Equals e hashCode

- Sempre sobrescreva o método *hashCode* quando você sobrescrever o método *equals*.
  - Implementação de *hashCode* incorreto é uma fonte de bugs difíceis de identificar
  - Impede o funcionamento apropriado em um conjunto de *collections* baseados em *hash*
  - Use os mesmos atributos do *equals* no método *hashCode*
- Este método não deve emitir *ClassCastException* nem *NullPointerException*.
- Não escreva um método *equals* complicado
- Não substitua o tipo *Object* por outro tipo na declaração de *equals*. (Coloque o *@Override*)
- Evite o uso de atributos que não existam em determinado ciclo de vida do objeto
- Hoje em dia as ferramentas (eclipse por exemplo) gera esses métodos para você a partir dos atributos selecionados

# Exercício Equals / hashCode

- Implemente o método equals e hashCode na classe conta
- Dica clique na classe com o botão direto-> source-> Generate Equals e hashCode
- Substitua a comparação no método transfer da classe AccountManager para usar o equals de Account

```
public class AccountManager {  
  
    public static void transfer(double value, Account origin, Account target) {  
  
        if(! origin.getId().equals(target.getId())) {  
  
            origin.withdraw(value);  
            target.deposit(value);  
        }  
    }  
}
```



# Collections

- Arrays não podem ser redimensionados
- Buscas são por índice ou percorrendo sequencialmente todo o array
- Collection
  - List
    - ArrayList
    - LinkedList
  - Set
    - HashSet
  - Map
    - HashMap

# List

Melhor desempenho na busca (get)

```
List<String> aList = new ArrayList<String>();

aList.add("um");
aList.add("dois");

for (String s : aList) {
    System.out.println(s);
}

String firstElement = aList.get(0);
```

Melhor desempenho na inserção e remoção de elementos

```
List<String> aList = new LinkedList<String>();

aList.add("um");
aList.add("dois");

for (String s : aList) {
    System.out.println(s);
}

String firstElement = aList.get(0);

System.out.println(firstElement);
return firstElement;
```

# List

```
List<String> aList = new ArrayList<String>();
```

```
aList.add("um");  
aList.add("dois");
```

```
for (String s : aList) {  
    System.out.println(s);  
}
```

```
String firstElement = aList.get(0);
```

```
List aList = new ArrayList();
```

```
aList.add("um");  
aList.add(16);
```

```
for (Object object : aList) {  
  
}
```

```
String firstElement = (String)aList.get(0);  
Integer secpndElement = (Integer)aList.get(1);
```

# Set

```
Set<String> s = new HashSet<String>();  
  
s.add("um");  
s.add("dois");  
s.add("dois");  
  
for (String s1 : s) {  
    System.out.println(s1);  
}  
  
System.out.println(s.size());
```

# Map

```
Map<String, Integer> m = new HashMap<>();
```

```
m.put("1", 1);
```

```
m.put("2", 2);
```

```
Integer one = m.get("1");
```

```
Integer two = m.get("2");
```

```
m.containsKey("1");
```

```
m.containsValue(1);
```