

Programación Orientada a Objetos

Excepciones

CEIS

2020-2

Agenda

Excepciones

- Introducción

- Definición

- Tres momentos

- Clases de excepciones

- Comportamiento

Buenas prácticas

- Refactorización

- Documentación

Ejemplos

- Polinomio

- Batalla Naval

Agenda

Excepciones

- Introducción

- Definición

- Tres momentos

- Clases de excepciones

- Comportamiento

Buenas prácticas

- Refactorización

- Documentación

Ejemplos

- Polinomio

- Batalla Naval

Introducción

Concepto

Una excepción es un mensajero que indica que algo fuera de lo normal está pasando

¿Qué tan fuera de lo normal?

Introducción

Concepto

Una excepción es un mensajero que indica que algo fuera de lo normal está pasando

¿Qué tan fuera de lo normal?

- ▶ POCO

Algo posible

Un caso especial

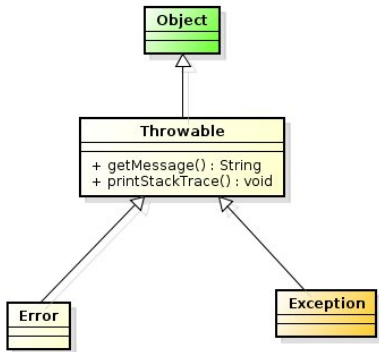
- ▶ MUCHO

Algo no esperado

Un error

Clase excepciones

¿Qué es?



The **Throwable** class is the superclass of all errors and exceptions in the Java language.

An **Error** is a subclass of **Throwable** that indicates serious problems that a reasonable application should not try to catch. Most such errors are abnormal conditions.

The class **Exception** and its subclasses are a form of **Throwable** that indicates conditions that a reasonable application might want to catch.

Introducción

¿Qué se debe hacer para trabajar con nuestras excepciones?

¿Qué se puede hacer con todas ellas?

Introducción

¿Qué se debe hacer para trabajar con nuestras excepciones?

1. Definir una clase propia para las excepciones
2. Crear el objeto correspondiente a la excepción

¿Qué se puede hacer con todas ellas?

1. Lanzarlas
2. Propagarlas
3. Atenderlas

Agenda

Excepciones

Introducción

Definición

Tres momentos

Clases de excepciones

Comportamiento

Buenas prácticas

Refactorización

Documentación

Ejemplos

Polinomio

Batalla Naval

Clase excepciones

¿Cómo se define una nueva clase excepción?

```
public class MissingValueException extends Exception {  
    // We've added a constructor ...  
    public MissingValueException(String message) {  
        // ... which simply invokes the base class constructor.  
        super(message);  
    }  
}
```

¿Cómo se crea un objeto excepción?

```
new MissingValueException("A student's name cannot be blank");  
:  
)
```

```
new Exception("We want to report an error");
```

Agenda

Excepciones

Introducción

Definición

Tres momentos

Clases de excepciones

Comportamiento

Buenas prácticas

Refactorización

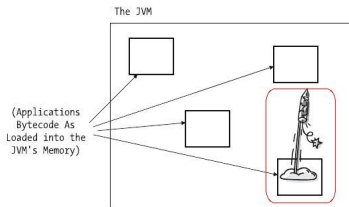
Documentación

Ejemplos

Polinomio

Batalla Naval

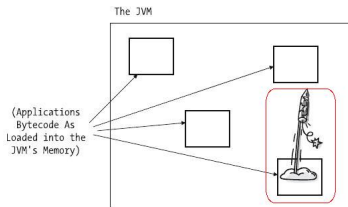
Lanzar



```
// We want to report an error if  
// the String that has been  
// passed in is blank.
```

```
if (n.equals("")) {  
    throw new MissingValueException(  
        "A student's name cannot be blank");  
} else {  
    name = n;  
}
```

Lanzar




```
// We want to report an error if  
// the String that has been  
// passed in is blank.
```

```
if (n.equals("")) {  
    throw new MissingValueException(  
        "A student's name cannot be blank");  
} else {  
    name = n;  
}
```

Refactorización

- La excepción “rompe” el código. ¿Qué sobraría?

Propagar



```
public void setName(String s) throws MissingValueException {  
    if (s.equals("")) {  
        throw new MissingValueException( "A student's name cannot be blank");  
    }  
    name = s;  
}  
  
public void upDate (String n String s) {  
    setName(n);  
    setSsn(s);  
    setMajor("UNDECLARED");  
}
```

¿Qué pasa?

Propagar



```
public void setName(String s) throws MissingValueException {  
    if (s.equals("")) {  
        throw new MissingValueException( "A student's name cannot be blank");  
    }  
    name = s;  
}
```

```
public void upDate (String n String s) throws MissingValueException {  
    setName(n);  
    setSsn(s);  
    setMajor("UNDECLARED");  
}
```

Debemos decidir que hacer con la excepción

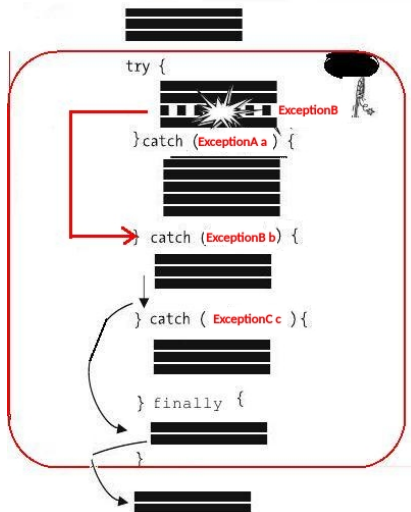
Atender



```
public static void main(String[] args) {  
    // Pseudocode.  
    Student s    ;  
  
    String name = read value from GUI;  
    try {  
        s.setName(name);  
  
    } catch (MissingValueException e) {  
        System.out.println(e.getMessage());  
        System.out.println("ID of affected student: " +  
                             s.getSsn());  
    }  
}
```

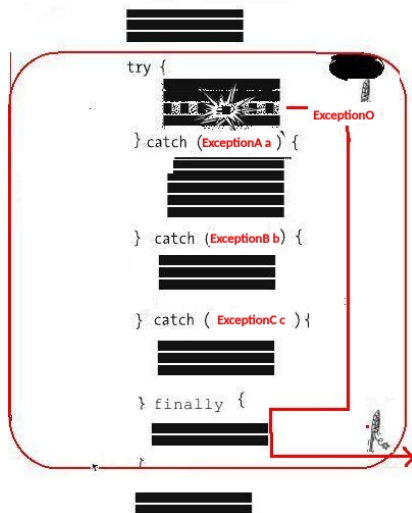

Bloque protegido

Excepción que se captura



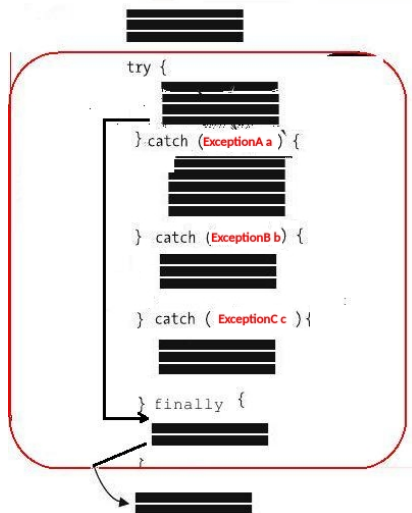
Bloque protegido

Excepción que no se captura



Bloque protegido

No hay excepcion



Agenda

Excepciones

Introducción

Definición

Tres momentos

Clases de excepciones

Comportamiento

Buenas prácticas

Refactorización

Documentación

Ejemplos

Polinomio

Batalla Naval

Clases de excepciones

API

java.lang

Class Exception

```
java.lang.Object
├── java.lang.Throwable
└── java.lang.Exception
```

All Implemented Interfaces:

[Serializable](#)

Direct Known Subclasses:

[AclNotFoundException](#), [ActivationException](#), [AlreadyBoundException](#), [ApplicationException](#), [AWTException](#), [BackingStoreException](#), [BadAttributeValueExpException](#), [BadBinaryOpValueExpException](#), [BadLocationException](#), [BadStringOperationException](#), [BrokenBarrierException](#), [CertificateException](#), [ClassNotFoundException](#), [CloneNotSupportedException](#), [DataFormatException](#), [DatatypeConfigurationException](#), [DestroyFailedException](#), [ExecutionException](#), [ExpandVetoException](#), [FontFormatException](#), [GeneralSecurityException](#), [GSSEException](#), [IllegalAccessException](#), [IllegalClassFormatException](#), [InstantiationException](#), [InterruptedException](#), [IntrospectionException](#), [InvalidApplicationException](#), [InvalidMidiDataException](#), [InvalidPreferencesFormatException](#), [InvalidTargetObjectTypeException](#), [InvocationTargetException](#), [IOException](#), [JMXException](#), [LastOwnerException](#), [LineUnavailableException](#), [MidiUnavailableException](#), [MimeTypeParseException](#), [NamingException](#), [NoninvertibleTransformException](#), [NoSuchFieldException](#), [NoSuchMethodException](#), [NotBoundException](#), [NotOwnerException](#), [ParseException](#), [ParserConfigurationException](#), [PrinterException](#), [PrintException](#), [PrivilegedActionException](#), [PropertyVetoException](#), [RefreshFailedException](#), [RemarshalException](#), [RuntimeException](#), [SAXException](#), [ServerNotActiveException](#), [SQLException](#), [TimeoutException](#), [TooManyListenersException](#), [TransformerException](#), [UnmodifiableClassException](#), [UnsupportedAudioFileException](#), [UnsupportedCallbackException](#), [UnsupportedFlavorException](#), [UnsupportedLookAndFeelException](#), [URISyntaxException](#), [UserException](#), [XAException](#), [XMLParseException](#), [XPathException](#)

Excepciones JAVA

Clases de excepciones

API

Overview Package **Class** Use Tree Deprecated Index Help

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#)

SUMMARY: NESTED | FIELD | [CONST](#) | [METHOD](#)

DETAIL: FIELD | [CONST](#) | METHOD

Java™ 2 Platform
Standard Ed. 5.0

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](#), [DOMException](#), [EmptyStackException](#), [EnumConstantNotPresentException](#), [EventException](#), [IllegalArgumentException](#), [IllegalMonitorStateException](#), [IllegalPathStateException](#), [IllegalStateException](#), [ImagingOpException](#), [IncompleteAnnotationException](#), [IndexOutOfBoundsException](#), [JMRuntimeException](#), [LSEException](#), [MalformedParameterizedTypeException](#), [MissingResourceException](#), [NegativeArraySizeException](#), [NoSuchElementException](#), [NullPointerException](#), [ProfileDataException](#), [ProviderException](#), [RasterFormatException](#), [RejectedExecutionException](#), [SecurityException](#), [SystemException](#), [TypeNotPresentException](#), [UndeclaredThrowableException](#), [UnmodifiableSetException](#), [UnsupportedOperationException](#)

Excepciones no chequeada

Clases de excepciones

Excepciones chequeadas: Necesario

```
public class Example {  
    public static void main(String[] args) {  
        // Pseudocode.  
        String name = read value from GUI;  
        Student s = new Student();  
        s.setName(name); ?  
        // etc.  
    }  
}
```

the following compiler error would arise on the line that attempts to invoke the setName method of Student s:

```
Unreported exception MissingValueException; must be caught or declared to be thrown  
s.setName(name);
```

Excepciones NO chequeadas: Innecesario

```
public class Professor {  
    // Details omitted.  
    public void methodY() {  
        // A NullPointerException is thrown here, but  
        // is NOT caught/handled.  
        // (Details omitted.)  
    }  
}
```

Agenda

Excepciones

Introducción

Definición

Tres momentos

Clases de excepciones

Comportamiento

Buenas prácticas

Refactorización

Documentación

Ejemplos

Polinomio

Batalla Naval

Comportamiento

Contexto

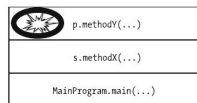
```
public class MainProgram {  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.methodX();  
        System.out.println("End main");  
    }  
}  
  
public class Student {  
    // Details omitted.  
    public void methodX() {  
        Professor p = new Professor();  
        p.methodY();  
        System.out.println("End methodX");  
    }  
}  
  
public class Professor {  
    // Details omitted.  
    public void methodY() {  
        // Details omitted ...  
        System.out.println("End methodY");  
    }  
}
```

Comportamiento. Caso 1.

Fuentes

```
public class MainProgram {  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.methodX();  
        System.out.println("End main");  
    }  
}  
  
public class Student {  
    // Details omitted.  
    public void methodX() {  
        Professor p = new Professor();  
        p.methodY();  
        System.out.println("End methodX");  
    }  
}  
  
public class Professor {  
    // Details omitted.  
    public void methodY() {  
        try { ... }  
        catch (NullPointerException e) { ... }  
        System.out.println("End methodY");  
    }  
}
```

Memoria

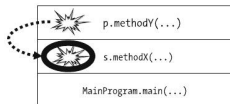


Comportamiento. Caso 2.

Fuentes

```
public class MainProgram {  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.methodX();  
    }    System.out.println("End main");  
}  
  
public class Student {  
    // Details omitted.  
    public void methodX() {  
        Professor p = new Professor();  
  
        // Exception handling is performed here.  
        try {  
            p.methodY();  
        }  
        catch (NullPointerException e) { ... }  
        System.out.println("End methodX");  
    }  
}  
  
public class Professor {  
    // Details omitted.  
    public void methodY() {  
        // A NullPointerException is thrown here, but  
        // is NOT caught/handled.  
        // (Details omitted.)  
        System.out.println("End methodY");  
    }  
}
```

Memoria



Comportamiento. Caso 3.

Fuentes

```
public class MainProgram {
    public static void main(String[] args) {
        Student s = new Student();

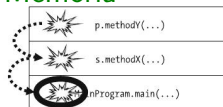
        // Exception handling introduced here.
        try {
            s.methodX();
        }
        catch (NullPointerException e) { ... }
        System.out.println("End main");
    }
}

public class Student {
    // Details omitted.
    public void methodX() {
        Professor p = new Professor();

        // We're not doing any exception handling
        // here, either.
        p.methodY();
        System.out.println("End methodX");
    }
}

public class Professor {
    // Details omitted.
    public void methodY() {
        // A NullPointerException is thrown here, but
        // is NOT caught/handled.
        // (Details omitted.)
        System.out.println("End methodY");
    }
}
```

Memoria

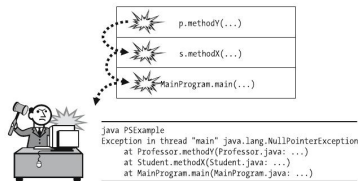


Comportamiento. Caso 4.

Fuentes

```
public class MainProgram {  
    public static void main(String[] args) {  
        Student s = new Student();  
        // We're not doing any exception handling  
        // here, either.  
        System.out.println("End main");  
    }  
}  
  
public class Student {  
    // Details omitted.  
    public void methodX() {  
        Professor p = new Professor();  
        // We're not doing any exception handling  
        // here, either.  
        p.methodY();  
        System.out.println("End methodX");  
    }  
}  
  
public class Professor {  
    // Details omitted.  
    public void methodY() {  
        // A NullPointerException is thrown here, but  
        // is NOT caught/handled.  
        // (Details omitted.)  
        System.out.println("End methodY");  
    }  
}
```

Memoria



Agenda

Excepciones

Introducción

Definición

Tres momentos

Clases de excepciones

Comportamiento

Buenas prácticas

Refactorización

Documentación

Ejemplos

Polinomio

Batalla Naval

Refactorización

La clase

```
public class MissingValueException extends Exception {  
    // We've added a constructor ...  
    public MissingValueException(String message) {  
        // ... which simply invokes the base class constructor.  
        super(message);  
    }  
}
```

Un método

```
public void setName(String s) throws MissingValueException {  
  
    if (s.equals("")) {  
        throw new MissingValueException( "A student's name cannot be blank");  
    }  
  
    name = s;  
}
```

Refactorización

La clase

```
public class MissingValueException extends Exception {  
    // We've added a constructor ...  
    public MissingValueException(String message) {  
        // ... which simply invokes the base class constructor.  
        super(message);  
    }  
}
```

Un método

```
public void setName(String s) throws MissingValueException {  
  
    if (s.equals("")) {  
        throw new MissingValueException( "A student's name cannot be blank");  
    }  
  
    name = s;  
}
```

- Mantener simple el código (método)

Refactorización

La clase

```
public class MissingValueException extends Exception {  
    // We've added a constructor ...  
    public MissingValueException(String message) {  
        // ... which simply invokes the base class constructor.  
        super(message);  
    }  
}
```

Un método

```
public void setName(String s) throws MissingValueException {  
    if (s.equals("")) throw new MissingValueException("A student's name cannot be blank");  
    name=s;  
}
```

- Mantener simple el código (método)

Refactorización

La clase

```
public class MissingValueException extends Exception {  
    // We've added a constructor ...  
    public MissingValueException(String message) {  
        // ... which simply invokes the base class constructor.  
        super(message);  
    }  
}
```

Un método

```
public void setName(String s) throws MissingValueException {  
    if (s.equals("")) throw new MissingValueException("A student's name cannot be blank");  
    name=s;  
}
```

- Encapsular los mensajes de usuario

Refactorización

La clase

```
public class MissingValueException extends Exception{  
    public static final String MISSING_NAME="A student's name cannot be blank";  
  
    public MissingValueException(String message){  
        super(message);  
    }  
}
```

Un método

```
public void setName(String s) throws MissingValueException {  
    if (s.equals("")) throw new MissingValueException(MissingValueException.MISSING_NAME);  
    name=s;  
}
```

- Encapsular los mensajes de usuario

Agenda

Excepciones

Introducción

Definición

Tres momentos

Clases de excepciones

Comportamiento

Buenas prácticas

Refactorización

Documentación

Ejemplos

Polinomio

Batalla Naval

Documentación

Código

```
/**
 * Change the student's name
 * @param s student's name
 * @throws MissingValueException if the new name is blank
 */
public void setName(String s) throws MissingValueException {
    if (s.equals("")) throw new MissingValueException(MissingValueException.MISSING_NAME);
    name=s;
}
```

HTML

setName

```
public void setName(java.lang.String s)
    throws MissingValueException
```

Change the student's name

Parameters:

s - student's name

Throws:

MissingValueException - if the new name is blank

Agenda

Excepciones

- Introducción

- Definición

- Tres momentos

- Clases de excepciones

- Comportamiento

Buenas prácticas

- Refactorización

- Documentación

Ejemplos

- Polinomio

- Batalla Naval

Polinomio

¿Qué pasaría?

```
public class Polinomio {  
    ....  
    private int[] coeficientes;  
  
    public Polinomio(int coeficientes[]) {  
        for (int i=0;i<(coeficientes.length);i++)  
            this.coeficientes[i]=(coeficientes[i]);  
    }  
  
    public int getCoficiente(int n){  
        return coeficientes[n];  
    }  
  
    public int getGrado (){  
        return coeficientes.length-1;..  
    }  
}
```

¿Excepciones?

Agenda

Excepciones

- Introducción

- Definición

- Tres momentos

- Clases de excepciones

- Comportamiento

Buenas prácticas

- Refactorización

- Documentación

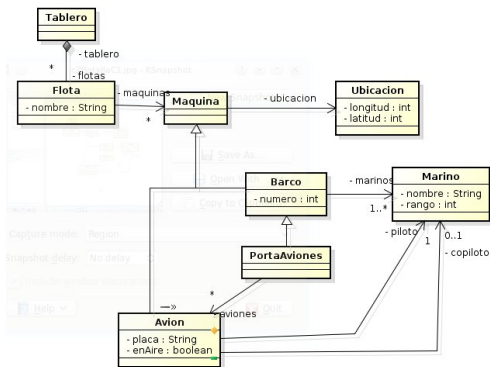
Ejemplos

- Polinomio

- Batalla Naval

Batalla Naval

Desarrollando



En Flota

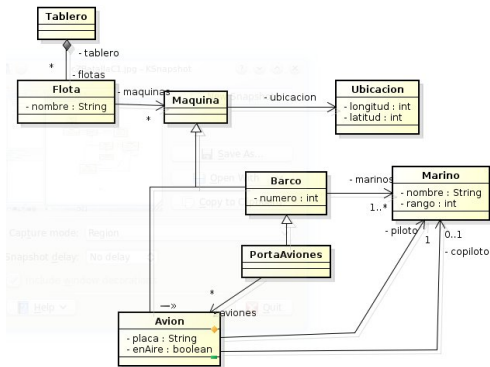
1. ¿El promedio de aviones en los portaaviones de una flota?

ASUMA TODO OK

`instanceof`

Batalla Naval

Desarrollando



En Flota

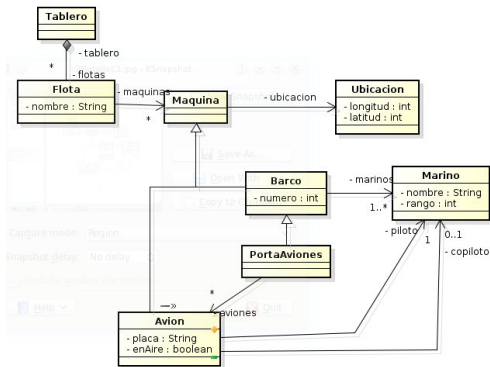
1. ¿El promedio de aviones en los portaaviones de una flota?

CASO IMPLICITO

Crear la clase para correspondiente a la excepción

Batalla Naval

Desarrollando



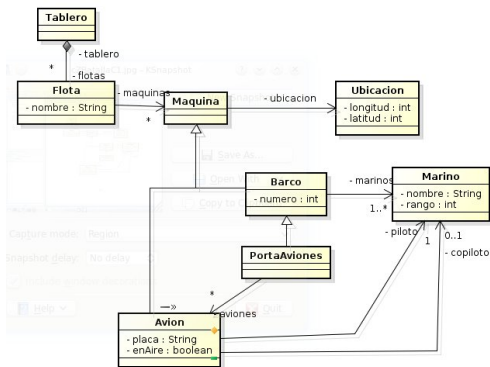
En Flota

1. ¿El promedio de aviones en los portaaviones de una flota?

CASO EXPLICITO. No se calcula si existe un porta-avión sin avión

Batalla Naval

Desarrollando



En tablero

1. La flota con mayor promedio de aviones en sus porta-aviones