Throwable

- The Throwable interface is meant to represent computational events that can interrupt the current computation
- Computation can occur after the event is handled
Exception

- Exceptions represent events that are meant to be treated.

- Whenever a method may trigger an exception, it is required that it declares so (modulo conformance). Except for RuntimeExceptions.
Runtime Exceptions

AnnotationTypeMismatchException, ArithmeticException,
ArrayStoreException, BufferOverflowException, BufferUnderflowException,
CannotRedoException, CannotUndoException, ClassCastException,
CMMException, ConcurrentModificationException, DOMException,
EmptyStackException, EnumConstantNotFoundException, EventException,
IllegalArgumentException, IllegalMonitorStateException,
IllegalPathStateException, IllegalStateException, ImagingOpException,
IncompleteAnnotationException, IndexOutOfBoundsException,
JMRuntimeException, ListException, MalformedParameterizedTypeException,
MissingResourceException, NegativeArraySizeException,
NoSuchElementException, NullPointerException, ProfileDataException,
ProviderException, RasterFormatException, RejectedExecutionException,
SecurityException, SystemException, TypeNotPresentException,
UndeclaredThrowableException, UnmodifiableSetException,

UnsupportedOperationException  ...
Error

• Meant to represent an unrecoverable error

• Can be recovered still...

• Example: AnnotationFormatError, AssertionError, AWTError, CoderMalfunctionError, FactoryConfigurationException, LinkageError, ThreadDeath, TransformerFactoryConfigurationException, VirtualMachineError
Throw, throws

- it is possible to throw an exception manually by using:
  \[ \text{throw an\_exception;} \]

- methods that may fail due to an exception (non-runtime) have to indicate it:

  \[ \text{public void m()} \text{ throws MyException\{...\}} \]
try...catch... finally

```java
try{
    ...
}
catch (MyException1 e){
    ...
}
catch (MyException2 e){
    ...
}
...
finally{
    ...
}
```
Java Basics: Part 4 - Streams

Manuel Oriol
• Streams are useful to apply to different inputs and outputs a single treatment with different results

• Basically, receive and send bytes

• Streams are responsible for handling the outer part of the communication
Input Streams

- InputStream regroup all objects that can receive information
- Can build readers on top of them, to handle the inner part of the communication
- InputStream API
Output Stream

- regroup all objects that can send information
- can build filters around them
- OutputStream API
Example: PrintStream

- System.out
- overloaded print/println/printf methods
- PrintStream API

note: printf is a variable arguments method...
Variable Argument Methods

• Arguments are automatically boxed into an array (http://java.sun.com/developer/JDCTechTips/2005/tt1018.html)

• Similar to C
Variable Arguments

Methods Example

```java
import java.util.*;
public class MyArgs {
    public static void main(String args[]) {
        method1("Hello", "World");
        method1(args);
        method2(12, 'a', "Hello", Math.PI, 1/3.0);
        method2(18, 94.0);
    }
    private static void method1(String... args) {
        System.out.println(Arrays.asList(args) + " // " + args.length);
    }
    private static void method2(int arg, Object... args) {
        System.out.println(Arrays.asList(args) + " / " + arg);
    }
}
```
Example: FileInputStream

- Reads from an file
- FileInputStream API
Reader

- For reading character streams...
- BufferedReader
- Reader API
  
  String readline()
System.out

• is a PrintStream
• can be changed (e.g. output in a file, socket...)
• by default is set to terminal/console output
System.in

- By default reads on the terminal/console
- Can be changed
- easier to build a BufferedReader on it
Using Streams for Keyboard Interactions

```java
PrintStream out =
    new PrintStream(new FileOutputStream("myfile.txt"));
out.println("My text");
out.close();
BufferedReader reader =
    new BufferedReader(new FileInputStream("myfile2.txt"));
// this time we append
out = new PrintStream(new FileOutputStream("myfile.txt"), true);
out.println("\t" + in.readLine());
```
Other Example: NoteTaker

```java
public static void main(String[] args) {
    String s;
    standard = new BufferedReader(new InputStreamReader(System.in));
    // checks arguments number
    if (args.length!=1) System.exit(0);
    // open the file name
    try {out = new FileOutputStream(args[0]);}
    catch (FileNotFoundException e){System.exit(0);}
    // users have to leave by using Control-C
    while(true){
        try {
            // read and write
            s=standard.readLine();
            out.write(s.getBytes());
            out.write("\n".getBytes());
        } catch (IOException e){
            System.out.println("I/O error");
            System.exit(0);
        }
    }
}
```
Basic Serialization

- ObjectOutputStream
- ObjectInputStream
Example OOS/OIS

```java
ObjectOutputstream out =
    new ObjectOutputStream(new FileOutputStream("myfile.txt"));
out.writeObject("test");
out.close();

ObjectInputstream ois =
    new ObjectInputstream(new FileInputStream("myfile.txt"));
String s;
// this time we read the object
s = (String) ois.readObject();
```
Socket streams

- SocketInputStream
- SocketOutputStream
Example Socket Streams

Socket s;
...
PrintStream out = new PrintStream(s.getOutputStream());
out.print("EOF");
...
BufferedReader reader = new BufferedReader(s.getInputStream());
// this time we read a line
String s = reader.readLine();
Java Basics:
Part 5 - Basic Swing
Model/View/Controller?

http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-
Model

- The model contains the application logic
- The model is typically an independent component
View

- This is the GUI itself
- An element of the GUI is a widget (e.g. menus, buttons, text fields...)
- Show the information
Controller

- This is the part that treats users interactions
- Controls the GUI to make it change when needed
- Controls the Model and changes it
Example of Model

Your application code (e.g. calculator component or model of whales travels)
Examples of View Components (1/3)

- JFrame:
  - creates a windows including a panel (getContentPane())
  - has to be shown (setVisible(true))
Examples of View Components (2/3)

- **JTable:**
  - A table of panels
  - takes an AbstractTableModel (Model component)
Examples of View Components (3/3)

• JButton
• JTextField
• JLabel
Controller

- ActionListener
Java Basics:
Part 6 - open questions
Green Cards Results

- DB + Java
- Web Services, XML
- Ant and Eclipse
- NetBeans
- GUI
- Networking
- Threads and Synchro
- New in 5.0 and 6.0?
- Secure Engineering in Java
- Patterns in Java
- J2ME
- Test Suites
- J2EE + JMS
- Eiffel vs Java
- JavaBeans
Green Cards Results

- DB + Java
- Web Services, XML
- Ant and Eclipse
- NetBeans
- GUI
- Networking
- Threads and Synchro
- New in 5.0 and 6.0?
- Secure Engineering in Java
- Patterns in Java
- J2ME
- Test Suites
- J2EE + JMS
- Eiffel vs Java
- JavaBeans
Green Cards Results

- DB + Java
- Networking
- Ant and Eclipse
- Threads and Synchro
- NetBeans
- New in 5.0 and 6.0?
- GUI
- Secure Engineering in Java
- XML
- Patterns in Java
- J2ME
- Test Suites
- J2EE & JMS
- Eiffel vs Java
- JavaBeans