Generic classes

- Declaration
- Use
- Annotations
• add generic types between angle brackets:

```java
public class MyStack<E,F>{
    E item;
}
```
Generic Classes Use

MyStack<Integer> ms =
    new MyStack<Integer>();
Generic Classes
Annotations

• public int count(MyClass<?> mc){...}

• public int count(MyClass<? extends String> mc){...}

• public int count(MyClass<? extends String & Countable> mc){...}
Why not subtype generics

```java
MyStack<Integer> myStack=new MyStack<Integer>();

public static void add(Stack<Object> st, Object o){
    st.add(o);
}
```
Example: MySmallStack

```java
public void add(E value){
    Element<E> e;
    e = new Element<E>(value);
    e.next = this.first;
    this.first = e;
}
...
MySmallStack<Integer> myStack =
    new MySmallStack<Integer>();
myStack.add(1);
```
Nested Classes

• Classes that are declared in a file, along another one

• If outside a class: static

• If inside, it depends... they can be inner classes
Inner Classes

- A particular type of nested classes
- Anonymous
- Method-scopes
- Class scope
Anonymous Inner Classes

• Declared at instanciation

```java
public void wave(){
    Object o = new Object(){
        public Object clone(){
            return this;
        }
    };
    ...
}
```
Method Inner classes

- declared in a method
- visible only in the method

```java
// method declaration
public void wave() {
    class Hello2 extends HelloWorld {
    }
    ...
}
```
Class Inner Classes

- Declared in the class
- visibility decides on access control outside of the class

```java
public class HelloWorld extends Object{
    class Hello2 extends HelloWorld{
    }
    // method declaration
    public void wave(){
      ...
    }
}

HelloWorld.Hello2 h=new HelloWorld.Hello2();
```
Some interesting classes

- Object
- String
- System
- Vector
Object

• See API
String

• See API
System

• See APIs
Vector

• See APIs
Java Basics
Part 3 - Exceptions
Manuel Oriol, April 13th, 2006
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, EnumConstantNotPresentException, EventException, IllegalArgumentException, IllegalMonitorStateException, IllegalPathStateException, IllegalStateException, ImagingOpException, IncompleteAnnotationException, IndexOutOfBoundsException, JMRuntimeException, LSException, 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, FactoryConfigurationError, LinkageError, ThreadDeath, TransformerFactoryConfigurationError, VirtualMachineError
Throw, throws

• it is possible to throw an exception manually by using: throw an_exception;

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

    public void m() throws MyException{...}
try...catch... finally

try{
...
} catch (MyException1 e){...}
catch (MyException2 e){...}
...
finally{...}