

Unit Testing and Error Handling



Unit testing
and test automation
within BlueJ

Overview

- Unit testing
- Test automation
- Error Handling in Java

Unit testing

- Each unit of an application may be tested
 - Method, class, module (package in Java)
- Can (should) be done during development
 - Finding and fixing early lowers development costs (e.g. programmer time)
 - A test suite is built up

Testing fundamentals

- Understand what the unit should do – its *contract*
 - You will be looking for violations
 - Use positive tests and negative tests
- Test *boundaries*
 - Zero, One, Full
 - Search an empty collection
 - Add to a full collection

Unit testing within BlueJ

- Objects of individual classes can be created
- Individual methods can be invoked
- Inspectors provide an up-to-date view of an object's state
- Explore through the *glossary* project

Test automation

- Good testing is a creative process, but thorough testing is time consuming and repetitive
- Use of a *test rig* or *test harness* can relieve some of the burden
 - Classes are written to perform the testing
 - Creativity focused in creating these



Error Handling in Java

- Main concepts to be covered:
- Defensive programming
 - Anticipating that things could go wrong
- Exception handling and throwing
- Error reporting

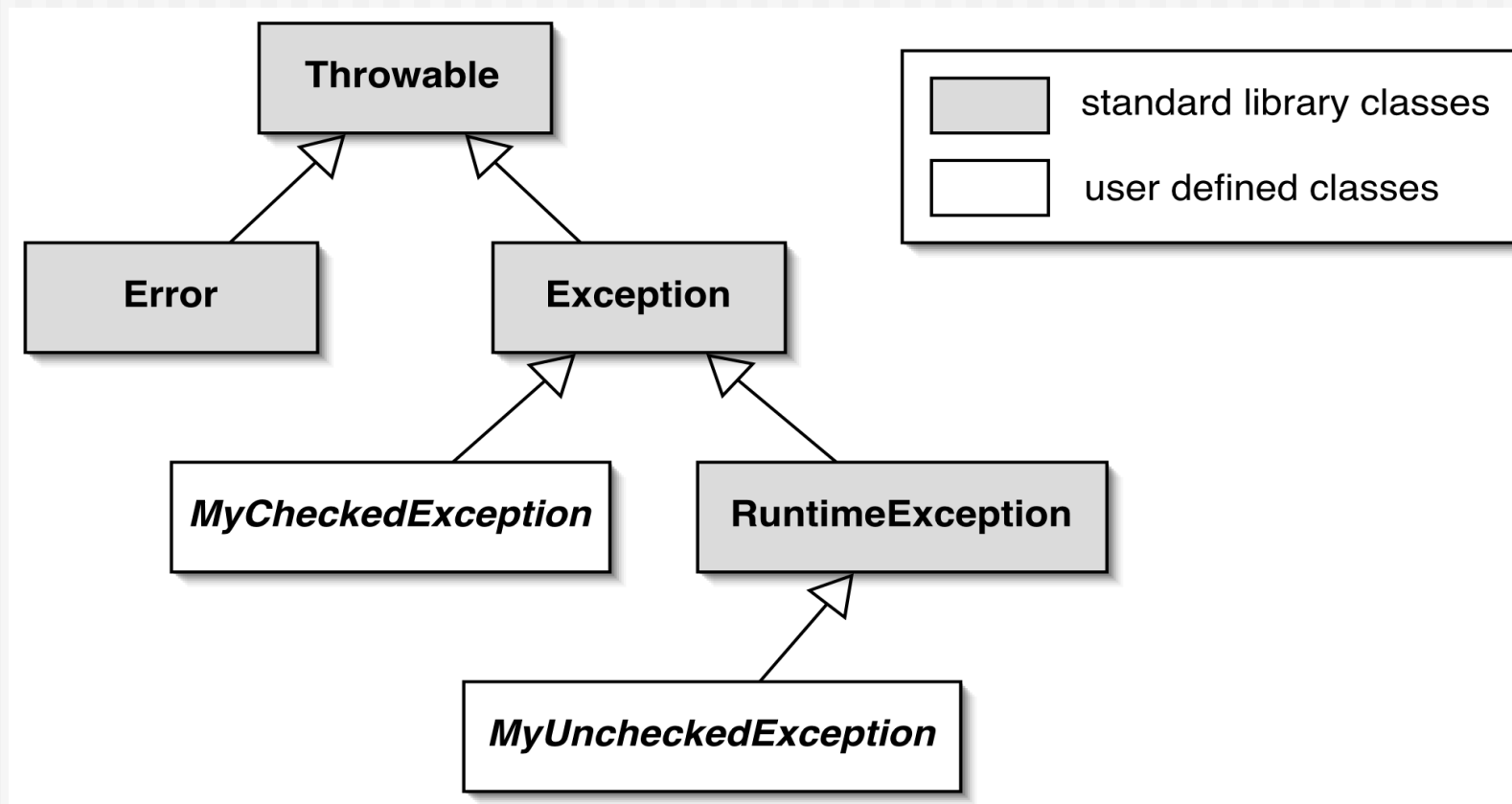
Argument values

- Arguments represent a major ‘vulnerability’ for an object
 - Constructor arguments initialize state
 - Method arguments often contribute to behavior
 - User inputs arguments for methods and constructors
- Argument checking is one defensive measure
 - Attempt recovery on error to avoid program failure
 - Or allow to fail, but inform about the reason

Exception-throwing principles

- A special language feature in Java
- No ‘special’ return value needed
- Errors cannot be ignored in the client
 - The normal flow-of-control is interrupted
- Specific recovery actions are encouraged
- If the exception is thrown:
 - The throwing method finishes prematurely
 - No return value is returned
 - Control does not return to the client’s point of call
 - The client cannot carry on regardless
 - But the client may ‘catch’ an exception

The exception class hierarchy



Unchecked Exceptions

- Unchecked exceptions are type of exceptions whose use will not require checks from the compiler
- Easy to use, they all are subclasses of `RuntimeException` class from `java.lang` package

```
if (key == null) {  
    throw new NullPointerException(  
        "Empty key passed to Details");  
}
```

Checked exceptions

- Require extra checks from the compiler
- Require **throws** clause and **try** statements

```
public void saveToFile(String destinationFile)  
                    throws IOException
```

- Clients catching an exception must protect the call with a **try** statement:

```
try {  
    Protect one or more statements here  
}  
catch (Exception e) {  
    Report and recover from the exception here  
}
```

The try statement example

1. Exception thrown from here

```
try {  
    addressbook.saveToFile(filename);  
    tryAgain = false;  
}  
catch(IOException e) {  
    System.out.println("Unable to save to " + filename);  
    tryAgain = true;  
}
```

2. Control transfers to here

Catching multiple exceptions

```
try {
    ...
    ref.process();
    ...
}
catch (EOFException e) {
    // Take action on an end-of-file exception.
    ...
}
catch (FileNotFoundException e) {
    // Take action on a file-not-found exception.
    ...
}
```

The finally clause

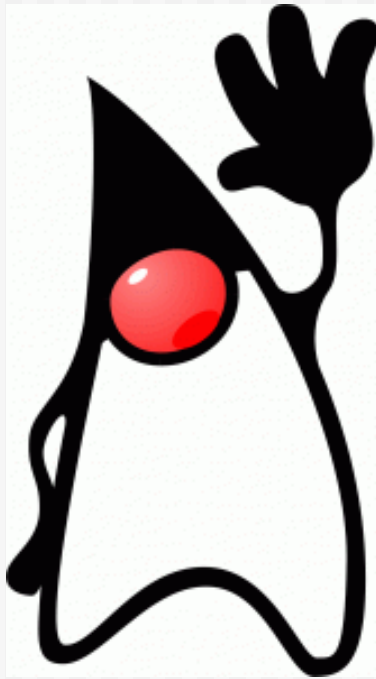
- A finally clause is executed even if a return statement is executed in the try or catch clauses
- A uncaught or propagated exception still exits via the finally clause

```
try {  
    Protect one or more statements here  
}  
catch (Exception e) {  
    Report and recover from the exception here  
}  
finally {  
    Perform any actions here common to whether or not  
    an exception is thrown  
}
```

Key Points (so far)

- Testing is an important part of software development
- Can be automated
- Error handling in Java realized via the Exceptions mechanism
- Can create your own Exception classes
- Can anticipate Exceptions and recover from errors

Using Java without BlueJ



How to write, compile
and run Java programs
outside BlueJ

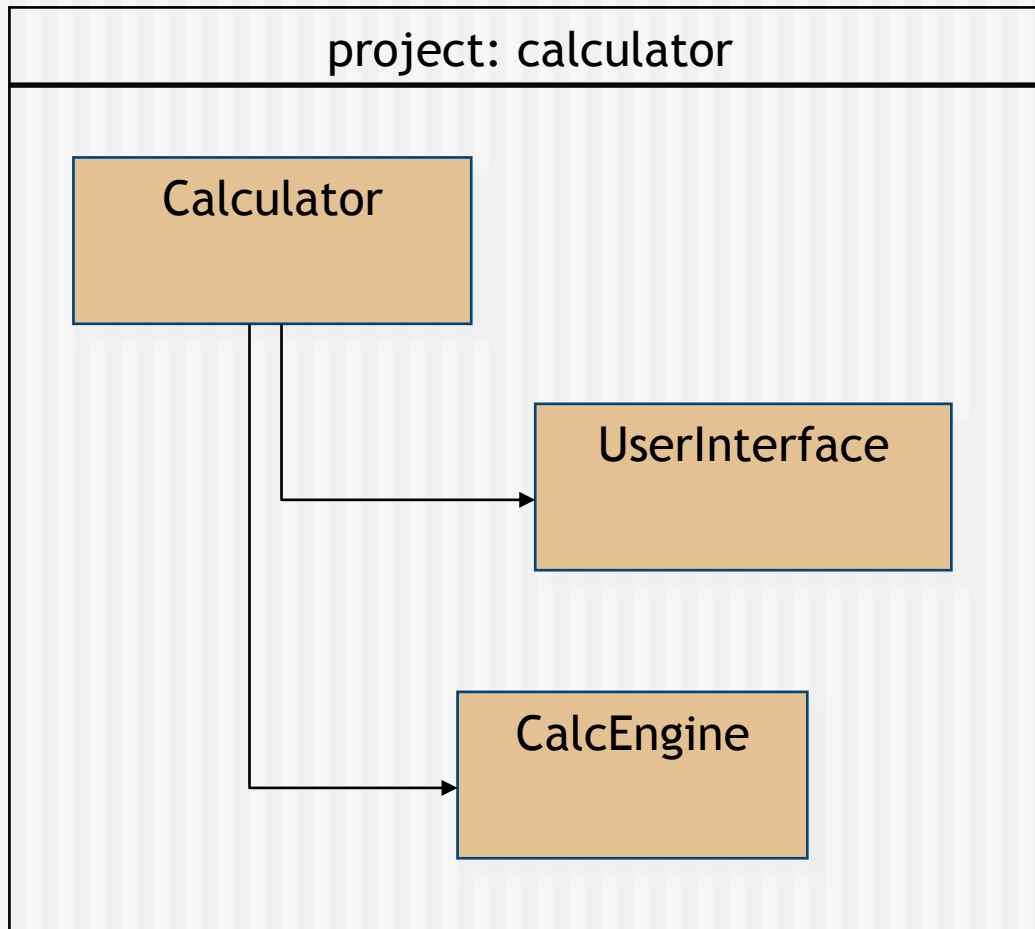
Overview

- BlueJ and standard Java
- Writing, compiling, testing and running Java code
- Java IDEs

BlueJ projects

- A BlueJ project is stored in a directory on disk
- A BlueJ package is stored in several different files
- Some files store the source code, some store the compiled code, some store additional information
- BlueJ uses standard Java format for some files and adds some additional files with extra information

The BlueJ directory structure



c:\bluej\calculator\
bluej.pkg
bluej.pkh
Calculator.java
Calculator.class
Calculator.ctxt
UserInterface.java
UserInterface.class
UserInterface.ctxt
CalcEngine.java
CalcEngine.class
CalcEngine.ctxt

The BlueJ file structure

- bluej.pkg - the package file
 - Contains information about classes in the package
 - One per package
- bluej.pkh - backup of the package file
- *.java - standard Java source file (text)
 - One per class
- *.class - standard Java code file
 - One per class
- *.ctxt - BlueJ context file
 - Contains extra information for a class
 - One per class

Standard Java files

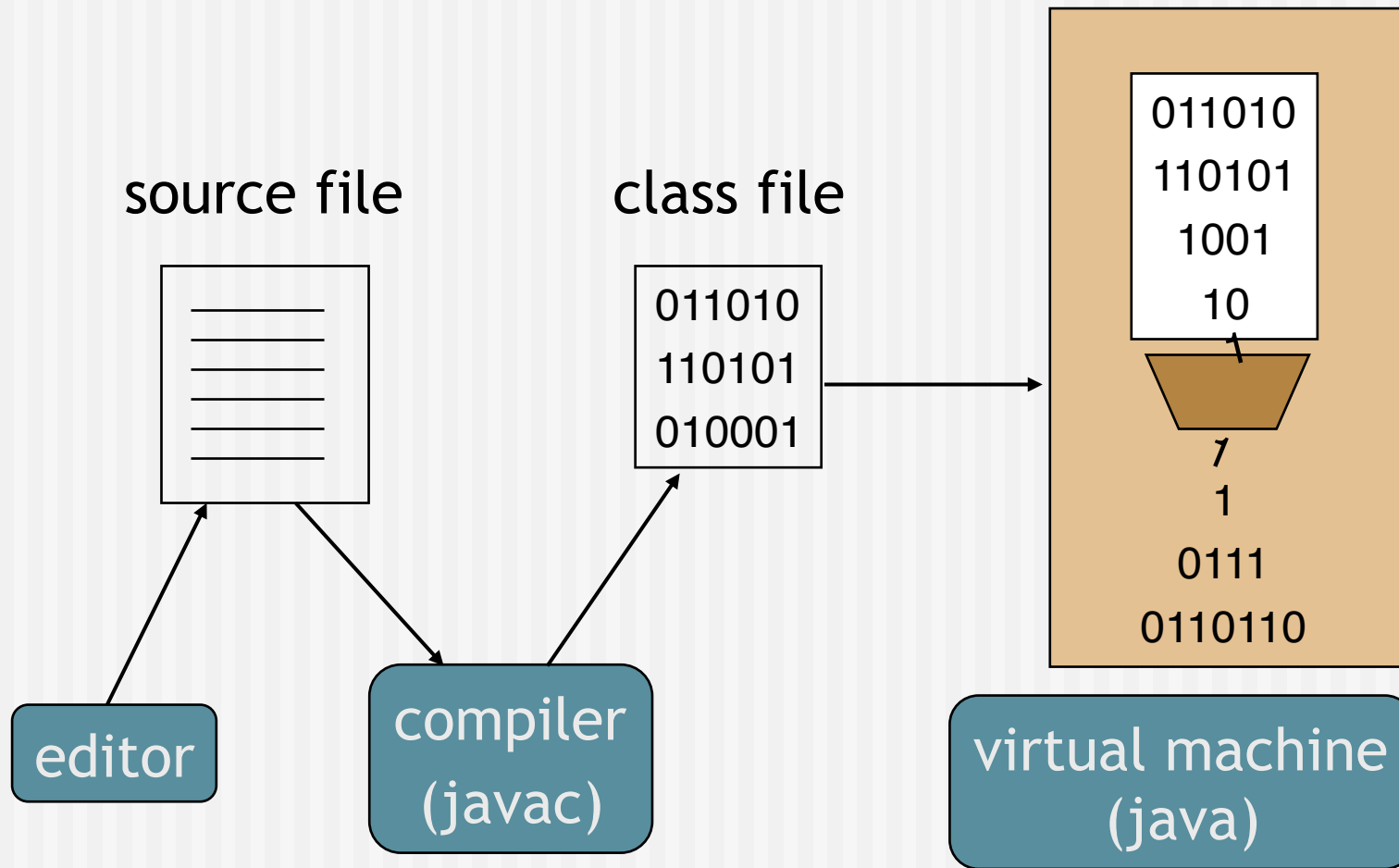
source files: `*.java`

Java source files contain the source code in readable form, as typed in by the programmer

class files: `*.class`

Java class files contain byte code (a machine readable version of the class). They are generated by the compiler from the source file.

The edit-compile-execute cycle



Editing

- A file can be edited in any text editor
 - Notepad, eMacs, jEdit, PFE, vi, ...
- Don't use Word: by default, Word does not save in text format
- Make sure to save with a `.java` filename before compiling!

Command line invocation

- Compilation and execution of Java in JDK are done from a command line
- On Microsoft systems: DOS shell
- On Unix: Unix shell
- Must make sure that the commands for compiler and runtime are in the command path

Compiling

- Name of the JDK compiler: `javac`
- To invoke:
`javac <source name>`
- compiles `<source name>` and all classes it depends on
- Example:
`cd C:\bluej\zuul`
`javac Game.java`

Error messages

```
C:\bluej\zuul> javac Game.java
Game.java:22: ';' expected.
    private Parser parser
                        ^
1 error
C:\bluej\zuul>
```

The programmer has to open the file in the editor, find the line number, fix the error and recompile.

Execution

- `C:\bluej\zuul> java Game`
- “java” starts the Java virtual machine
- The named class is loaded and execution is started
- Other classes are loaded as needed
- Only possible if class has been compiled

Problem: Execute what?

- If we try:

```
C:\bluej\zuul> java Game  
Exception in thread "main"  
java.lang.NoSuchMethodError: main
```

- The problem: how does the system know which method to execute?

The main method

- The answer: The java system always executes a method called **main** with a certain signature:

```
public static void main(String[] args)
{
    ...
}
```

- For this to work, such a method **must exist!**

The main method *(cont.)*

- “main” must exist
- “main” must be *public*
- “main” must be *static* (class method)
- “main” must have a *String array* parameter
- Only “main” can be invoked

main method - example

```
public static void main(String[] args)
{
    Game game = new Game();
    game.play();
}
```

- The main method should
 - create an object
 - call the first method

Testing

- To test, test drivers must be written
- All test method calls must be written into a test method
- All relevant parameter combination must be tested
- If tests depend on the results of earlier tests, the test driver must be edited and recompiled
- The test driver must create the objects

Key Points

- You can write, edit and execute Java code in many specialized *integrated development environments* or *IDE* (for example, Eclipse, Netbeans, JCreator)
- Different IDEs suit different programming needs
- You can also write and edit your Java code in a simple text editor and compile and running using DOS(PC) or Shell terminals (Unix)