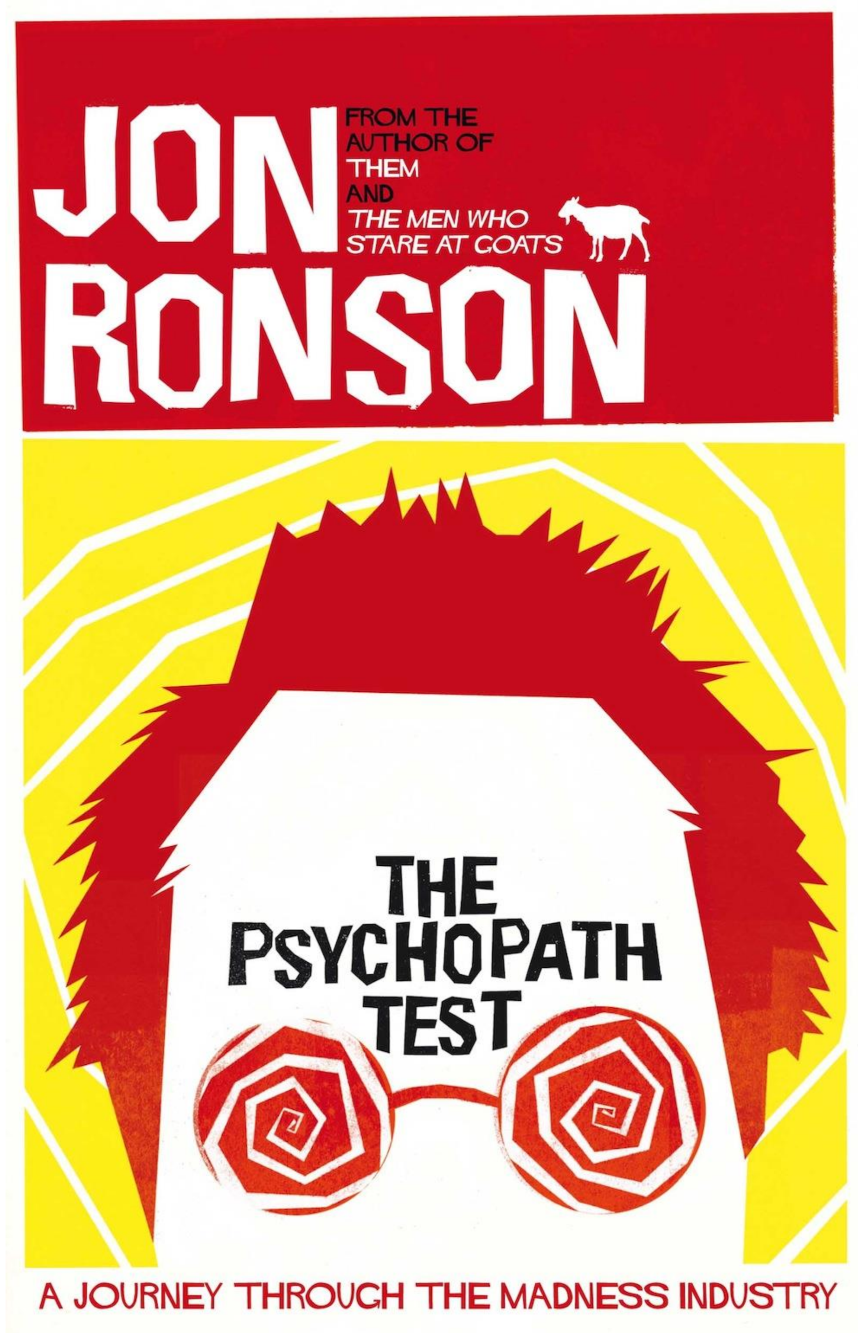
Java Programming 2
Annotations

Mary Ellen Foster

MaryEllen.Foster@glasgow.ac.uk

"Contemplating orange juice" by Hannah Webb
https://flic.kr/p/bkqXSy


JON RONSON
FROM THE AUTHOR OF THEM AND THE MEN WHO STARE AT GOATS
THE PSYCHOPATH TEST
A JOURNEY THROUGH THE MADNESS INDUSTRY

"Always code as if the [person] who ends up maintaining your code will be a violent psychopath who knows where you live. Code for readability."

John F. Woods

Annotations

A form of **metadata** – provide data about a program that is not part of the program itself

Have no direct effect on the operation of the code they annotate

Uses

- **Information for the compiler:** detect errors, suppress warnings
- **Compile-time processing:** use annotations to generate code/XML/etc
- **Runtime processing:** some annotations are available

Format of an annotation

Start with an @ sign

```
@Override
public String toString() { … }
```

Refer to the element following them

Must appear **outside** comments

May have arguments inside parens – if no arguments, parens can be omitted

```
@SuppressWarnings("unchecked")
void myMethod() { … }
```

Annotation locations

Generally, applied to declarations (classes, fields, methods, etc.)

Conventionally, each annotation appears on its own line

As of Java 8, annotations can also be applied to the use of types

- Ensures stronger type checking
- Not built into Java itself, but downloadable packages exist
  - *E.g., http://types.cs.washington.edu/checker-framework/*

```
@NonNull String str; // Won't work without external package
```

Useful predefined annotations

`@Deprecated`

Marks code as “deprecated” – i.e., still included but use is discouraged

`@Override`

Indicates that the labelled method must override a superclass method

`@SuppressWarnings`

Disables particular compiler warnings

All are defined in java.lang (e.g., `java.lang.Override`)

Use of @Override

“Indicates that a method declaration is intended to override a method declaration in a supertype” (Javadoc)

Compiler produces an error message unless this is true

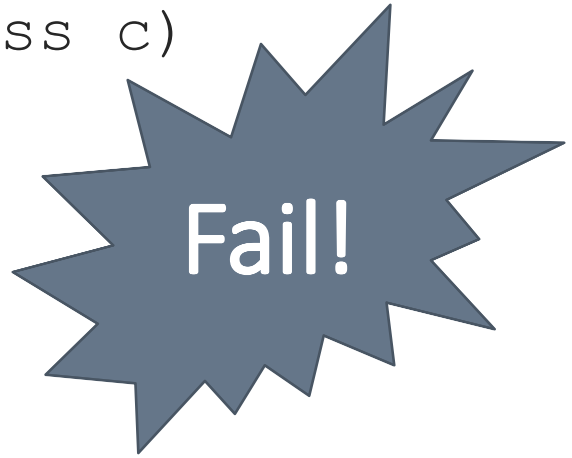
Automatically added by Eclipse whenever you override/implement methods

```
public class MyClass {
   @Override
   public boolean equals
      (MyClass c)

   {

      …

   }
}
```



Use of @SuppressWarnings

Tells compiler to suppress warnings that it would otherwise generate

Argument indicates category:

- **deprecation**: disable warning on use of deprecated method
- **unchecked**: disable warning on use of non-generic code

Full set of Eclipse warnings:

- http://help.eclipse.org/mars/index.jsp?topic=/org.eclipse.jdt.doc.user/tasks/task-suppress_warnings.htm

Should be attached to innermost element where they apply

- Do not disable warnings on a whole class if they are needed on one method!

@SuppressWarnings example

```
@SuppressWarnings(“unchecked”)
public void doSomethingOldFashioned() {
    ArrayList list = new ArrayList();
    list.add (“One”);
    list.add (2);
    list.add (3.0);
}
```

Adding annotations in Eclipse

Eclipse automatically adds @Override annotations to any auto-generated methods where it is relevant

- Implementing an interface
- Subclassing an abstract class
- Explicitly choosing “override/implement methods”

It often proposes a “quick fix” to suppress warnings when they occur

- Only do this if you are **REALLY REALLY SURE** the warning is not relevant!