NetBeans IDE e JUnit

Esercitazione in laboratorio

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

Bibliografia

- http://netbeans.org/kb/trails/java-se.html
- http://www.refactoring.com/
- http://wiki.netbeans.org/Refactoring
- http://ant.apache.org/manual/index.html
- http://svnbook.red-bean.com/
- http://www.junit.org/
- http://netbeans.org/kb/docs/java/junit-intro.html

NetBeans IDE

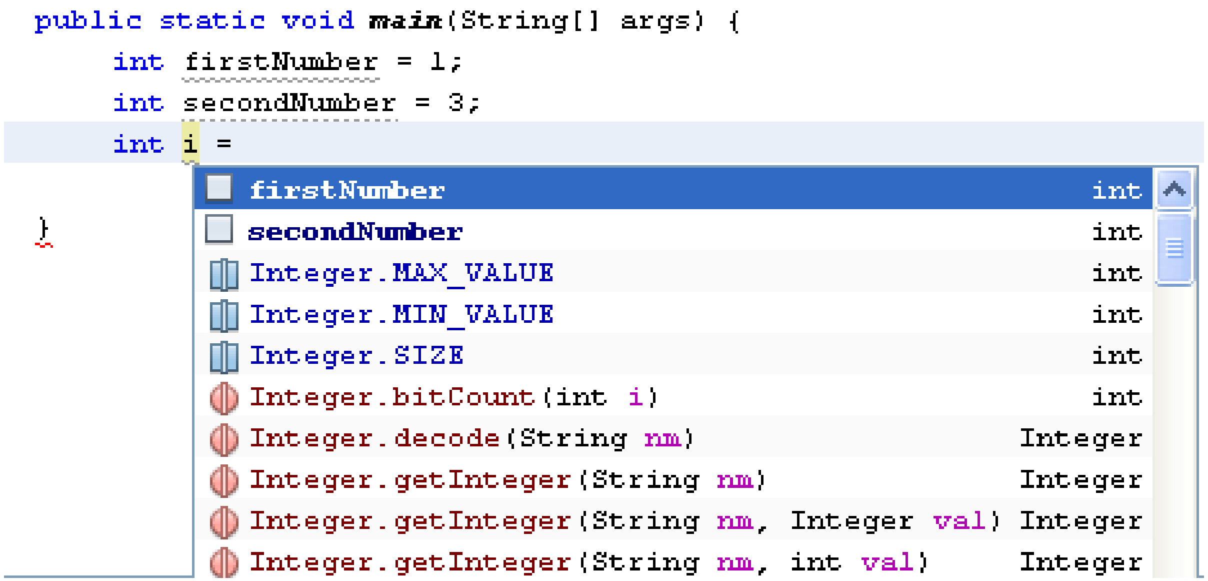
- Live Parsing
- Refactoring
- Smart Code Completion
- Jump to
  - Navigare all'interno del codice
  - Dall'errore alla linea
- Navigare per classi e interfacce

NetBeans IDE

- Swing gui buider
- Profiler
- Debugger
- Version control
- Connessioni a DB
- JavaEE
    - Web Frameworks
    - Web services

Code Completion

- Completamento automatico (ctrl-spazio)

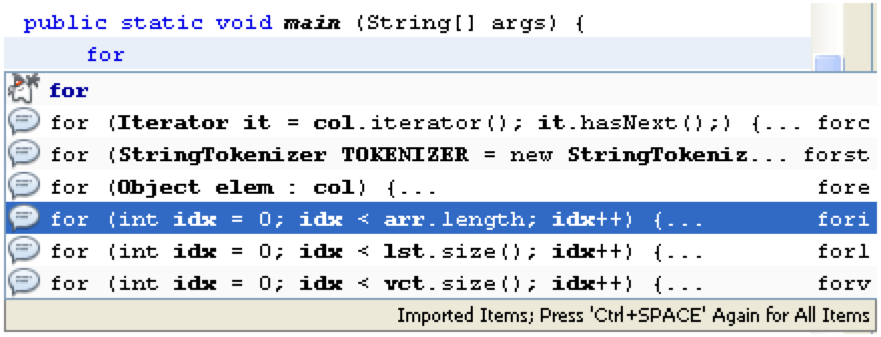


```
public static void main(String[] args) {
    int firstNumber = 1;
    int secondNumber = 3;
    int i =

}
```

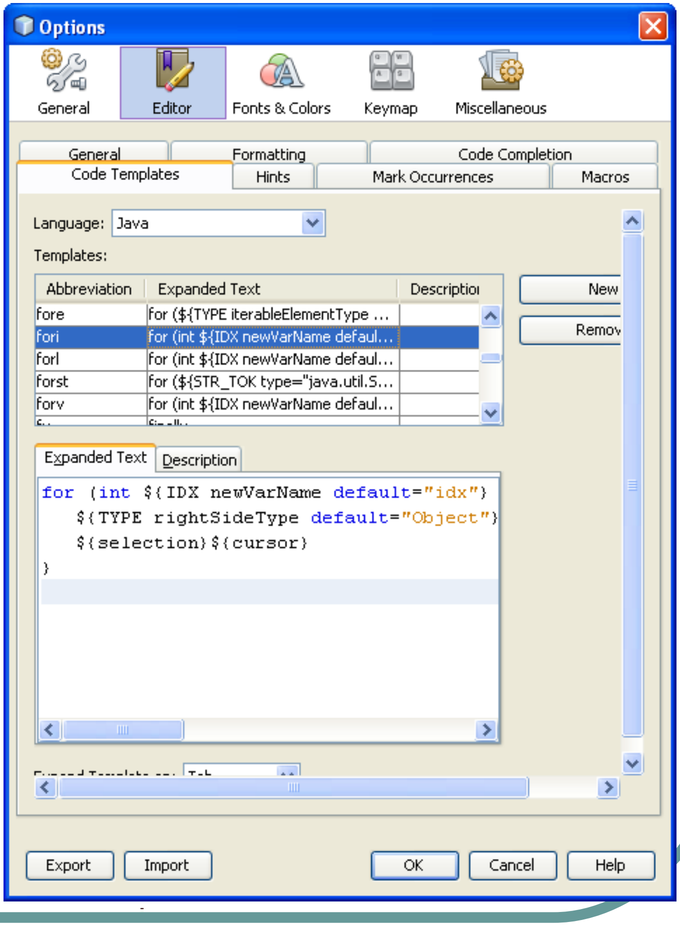
```
firstNumber                                    int
secondNumber                                   int
Integer.MAX_VALUE                              int
Integer.MIN_VALUE                              int
Integer.SIZE                                   int
Integer.bitCount(int i)                        int
Integer.decode(String nm)                  Integer
Integer.getInteger(String nm)              Integer
Integer.getInteger(String nm, Integer val) Integer
Integer.getInteger(String nm, int val)     Integer
```

Live Template



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

```
for
for (Iterator it = col.iterator(); it.hasNext();) {...   forc
for (StringTokenizer TOKENIZER = new StringTokeniz...  forst
for (Object elem : col) {...                            fore
for (int idx = 0; idx < arr.length; idx++) {...         fori
for (int idx = 0; idx < lst.size(); idx++) {...         forl
for (int idx = 0; idx < vct.size(); idx++) {...         forv
Imported Items; Press 'Ctrl+SPACE' Again for All Items
```



Language: Java

Templates:

Abbreviation	Expanded Text	Descriptio
fore	for (${TYPE iterableElementType ...	
fori	for (int ${IDX newVarName defaul...	
forl	for (int ${IDX newVarName defaul...	
forst	for (${STR_TOK type="java.util.S...	
forv	for (int ${IDX newVarName defaul...	

Expanded Text | Description

```
for (int ${IDX newVarName default="idx"}
   ${TYPE rightSideType default="Object"}
   ${selection}${cursor}
}
```

Suggerimento errori

- Netbeans compila e verfica la sintassi del codice “al volo”
- Ci indica subito i possibili errori
- E talvolta anche le probablili soluzioni
  - Es. aggiungere un import, modificare la visibilità

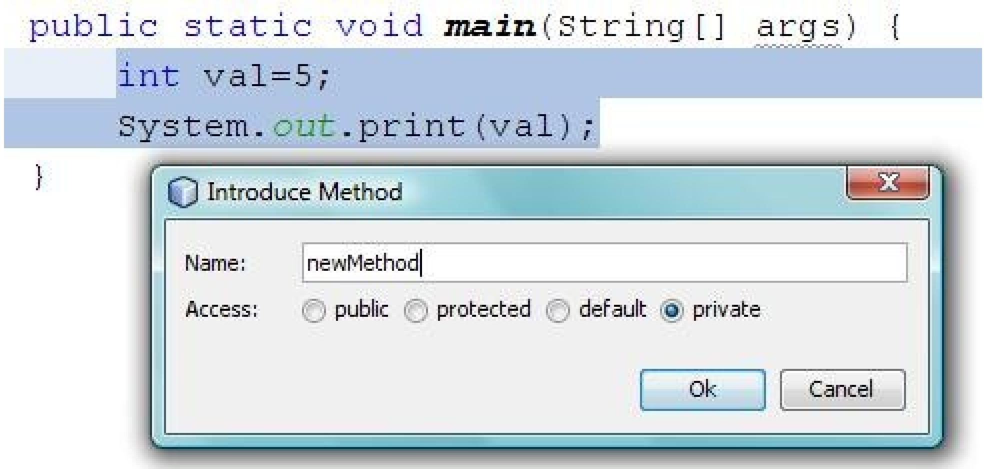
Refactoring

- “Any fool can write software thata a computer can understand. Good programmers write code that uman can understand
- “Refactoring is the process of changing a software system in such a way that it does not alter the external behaviour of the code yet improves its internal structure.”

Martin Fowler, Refactoring

- Perché?
  - Migliora il design del software
  - Rende il codice più comprensibile
  - Facilita la manutenzione
  - Aitua a trovare gli errori
  - Velocizza la programmazione
- Esempio: “rename” cambia il nome di un metodo o di una variabile in tutte le occorenze

Replace block of code with a method

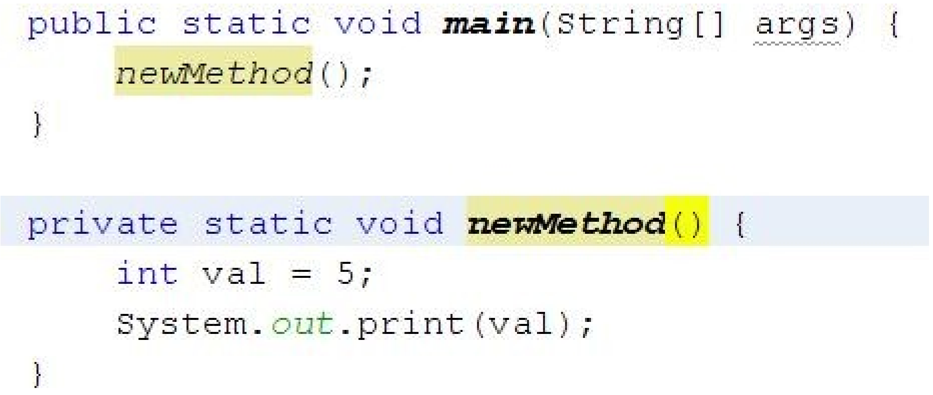


```
public static void main(String[] args) {
        int val=5;
        System.out.print(val);
}
```

Introduce Method

Name: newMethod

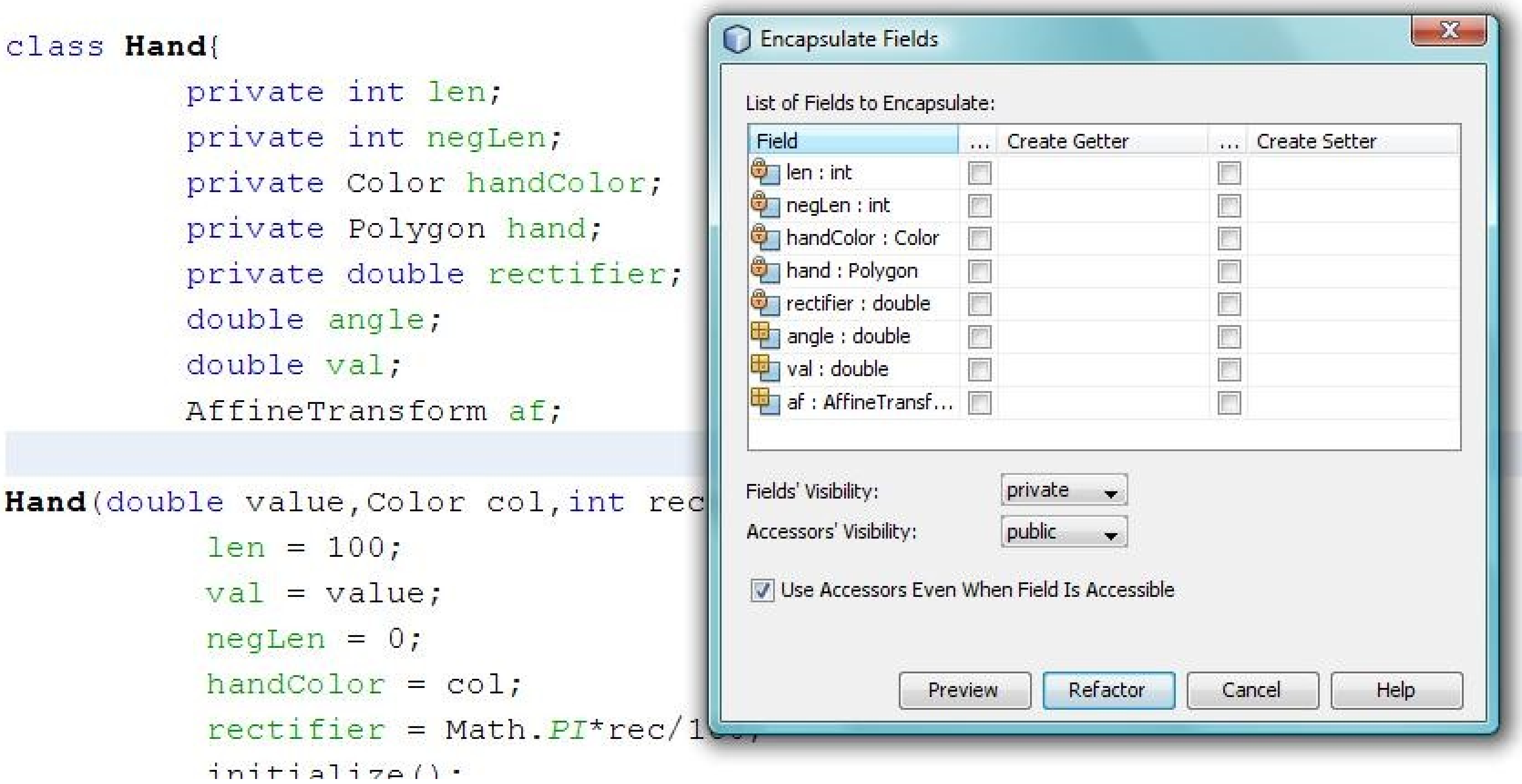
Access: ○ public ○ protected ○ default ● private



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

private static void newMethod() {
        int val = 5;
        System.out.print(val);
}
```

Encapsulate Fields



```
class Hand{
        private int len;
        private int negLen;
        private Color handColor;
        private Polygon hand;
        private double rectifier;
        double angle;
        double val;
        AffineTransform af;

Hand(double value,Color col,int rec
         len = 100;
         val = value;
         negLen = 0;
         handColor = col;
         rectifier = Math.PI*rec/1
         initialize();
```

Encapsulate Fields

List of Fields to Encapsulate:

Field	...	Create Getter	...	Create Setter
len : int	☐		☐	
negLen : int	☐		☐	
handColor : Color	☐		☐	
hand : Polygon	☐		☐	
rectifier : double	☐		☐	
angle : double	☐		☐	
val : double	☐		☐	
af : AffineTransf...	☐		☐	

Fields' Visibility: private

Accessors' Visibility: public

☑ Use Accessors Even When Field Is Accessible

Other frequently used refactoring techniques

- Move Class: Moves a class to another package or into another class. In addition, all source code in your project is updated to reference the class in its new location.
- Safely Delete: Checks for references to a code element and then automatically deletes that element if no other code references it.
- Change Method Parameters: Enables you to add parameters to a method and change the access modifier.
- Extract Interface: Creates a new interface from the selected public non-static methods in a class or interface.
- Extract Superclass: Creates a new abstract class, changes the current class to extend the new class, and moves the selected methods and fields to the new class.
- Move Inner to Outer Level: Moves an inner class or method one level up in hierarchy.

ANT

- Apache Ant is a Java-based build tool. In theory, it is kind of like *make*, without *make's* wrinkles." (ant.apache.org)
- Configurato in un file xml (build.xml)
- Estensioni scritte in java
- Platform-independent
- Compilazione, esecuzione, documentazione, archiviazione, test

Esempio base

```
<project default="hello">
   <target name="hello">
         <echo message="Hello, World"/>
   </target>
</project>
```

Execution build file :

```
$ ant
Buildfile: build.xml
   hello: [echo] Hello, World
   BUILD SUCCESSFUL
Total time: 2 seconds
```

Gerarchia

- Project: uno per file, ha un target di default
- Target: molti in un file, hanno dipendenze, contengono task
- Task: elemento eseguibile, realizzati in java, anche personalizzabili
- Properties: specificate nell'xml o caricate da un file

Esempio “hello.xml”

```
<project default="compile">
  <target name="compile">
    <javac srcdir="." />
  </target>

  <target name="jar" depends="compile">
    <jar destfile="hello.jar"
         basedir="."
         includes="**/*.class"
         />
  </target>
</project>
```

```
$ ant -f hello.xml jar
```

Task personalizzati

In tre passi (semplificato):

1. Creare una classe Java che estenda org.apache.tools.ant.Task
2. Scrivere un metodo setter per ogni attributo
3. Scrivere un metodo public void execute senza argomenti, che lanci una BuildException

Esempio finale (1/2)

```
<project name="MyProject" default="dist" basedir=".">
    <description> simple example build file </description>

    <property name="src" location="src"/>
    <property name="build" location="build"/>
    <property name="dist"  location="dist"/>

    <target name="init">
           <mkdir dir="${build}"/>
    </target>

    <target name="compile" depends="init”
       description="compile the source " >
           <javac srcdir="${src}" destdir="${build}"/>
    </target>
```

Esempio finale (2/2)

```
    <target name="dist" depends="compile"
               description="generate the distribution" >
           <mkdir dir="${dist}/lib"/>
           <jar jarfile="${dist}/lib/MyProject-${DSTAMP}.jar" basedir="${build}"/>
    </target>

    <target name="clean"
       description="clean up" >
           <delete dir="${build}"/>
           <delete dir="${dist}"/>
    </target>
</project>
```

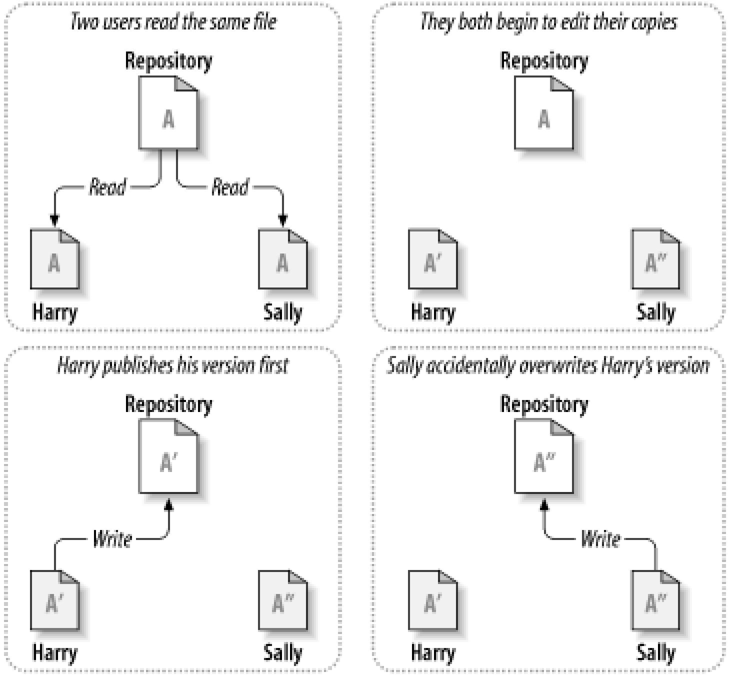
Gestire il codice

- Software multi-versione sviluppato da molti operatori. (def di ing.SW)
- Problemi:
  - Archiviazione comune
  - Gestione delle versioni
  - Spazio di lavoro
  - Diramazioni (famiglie)
- Soluzioni:
  - Mail con zip
  - Directory condivisa

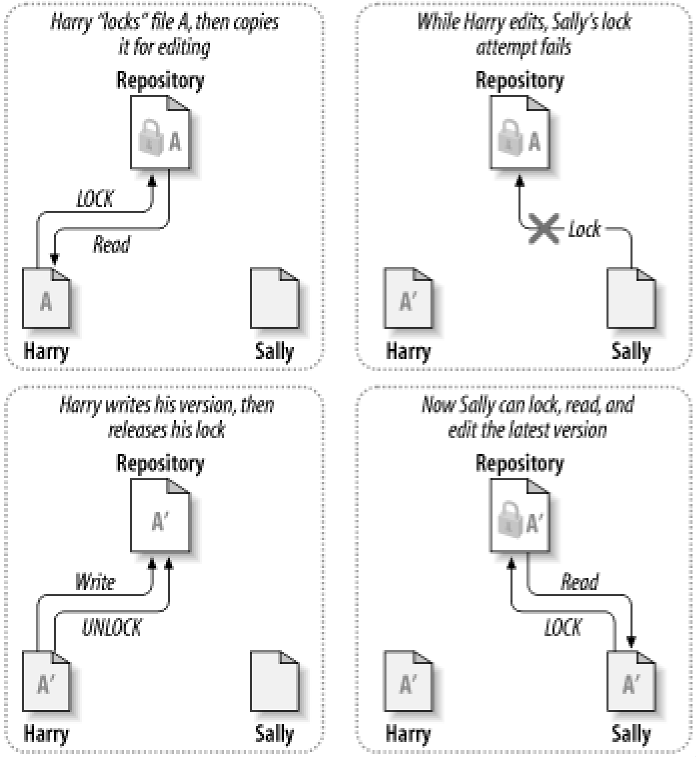
Subversion

- SVN is a source version control system. Si occupa di gestire un albero di file e directory nel tempo, tenendo traccia dei cambiamenti su ogni file.
- La struttura è depositata in un repository (incrementale).
- Avere a disposizione tutto lo “storico” di un file consente di recuperrare vecchie versioni dei file, e di esaminare la successione dei cambiamenti.
- SVN lavora in rete (client-server). Gli sviluppatori saricano le copie di lavoro (working copy), effettuano le modifiche e sottomettono I cambiamenti al server (commit).
- Consente di tornare (revert) a versioni del codice corrette, nel caso in cui siano state fatte modifiche errate.
- Consente lo sviluppo concorrente di un applicazione, gestendo la fusione di modifiche fatte da diverse persone.

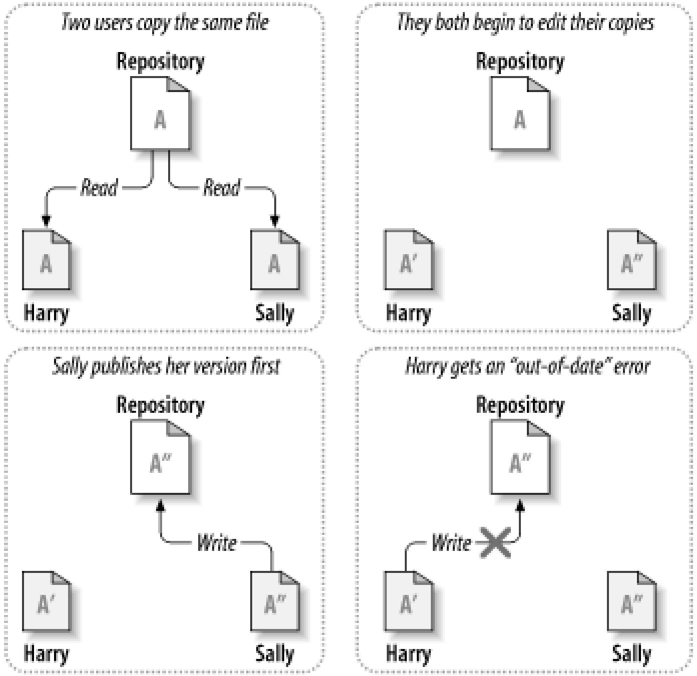
Il problema principale

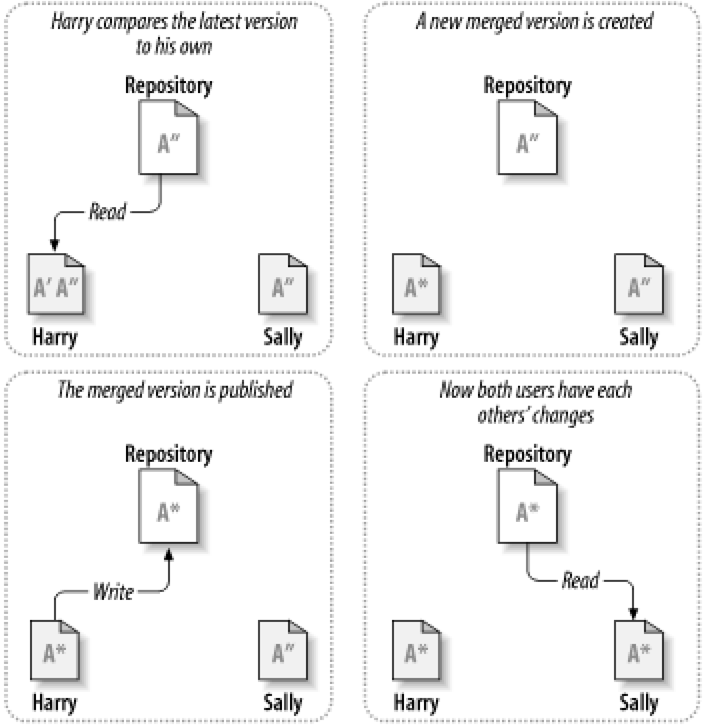

Two users read the same file
Repository
A
Read
Read
A
A
Harry
Sally
They both begin to edit their copies
Repository
A
A′
A″
Harry
Sally
Harry publishes his version first
Repository
A′
Write
A′
A″
Harry
Sally
Sally accidentally overwrites Harry's version
Repository
A″
Write
A′
A″
Harry
Sally

La soluzione blocca-modifica-sblocca (Lock-Modify-Unlock)

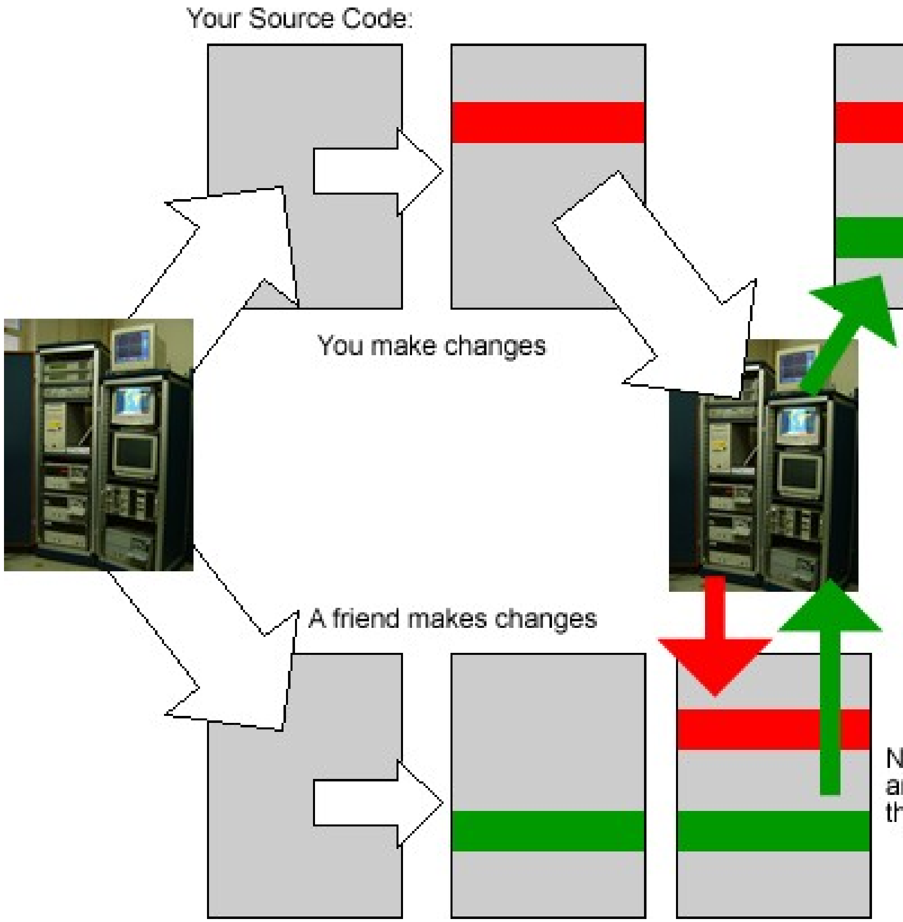

Harry "locks" file A, then copies it for editing
Repository
A
LOCK
Read
A
Harry
Sally
While Harry edits, Sally's lock attempt fails
Repository
A
Lock
A′
Harry
Sally
Harry writes his version, then releases his lock
Repository
A′
Write
UNLOCK
A′
Harry
Sally
Now Sally can lock, read, and edit the latest version
Repository
A′
Read
LOCK
A′
A′
Harry
Sally

Copia-modifica-fondi


Two users copy the same file
Repository
A
Read
Read
A
A
Harry
Sally
They both begin to edit their copies
Repository
A
A′
A″
Harry
Sally
Sally publishes her version first
Repository
A″
Write
A′
A″
Harry
Sally
Harry gets an "out-of-date" error
Repository
A″
Write
A′
A″
Harry
Sally


Harry compares the latest version to his own
Repository
A″
Read
A′ A″
A″
Harry
Sally
A new merged version is created
Repository
A″
A*
A″
Harry
Sally
The merged version is published
Repository
A*
Write
A*
A″
Harry
Sally
Now both users have each others' changes
Repository
A*
Read
A*
A*
Harry
Sally

Un ciclo di lavoro



Your Source Code:

You make changes

A friend makes changes

New Way: You both make changes and they are merged on-demand when you sync with the server.

SVN in breve

Check out a working copy from the repository. (Or, if you already have a working copy, run an "**update**" on your working copy.)

Make your changes, additions, etc. (write code!) Make sure to notify SVN of any new files or name/location changes of files.

Do an "**Update**" in your working copy. If there are others working on the code at the same time, and they commit changes to the repository, this will download those changes to your local copy. If there are conflicts (i.e. someone has changed something that conflicts with your local changes) you will be alerted and asked to fix those conflicts before you **commit**.

Commit your changes to the repository. Make sure you write an informative comment about your changes.

Project directory layout

trunk: filone principale di sviluppo (HEAD)

tags: versioni rilevanti e stabili

branches: diramazioni per sviluppo di particolari modifiche o derivazione di fmiglia.

Usare i tags!!! Non spercano spazio, sono solo simbolici.

Basic Subversion Commands (1/2)

- add: svn add foo
- cat: svn cat svn://foo
- checkout (co): svn co svn://host.com/dir/proj
- commit (ci): svn commit -m “log message”
- copy (cp): svn cp svn://foo svn://bar
- delete (rm): svn rm svn://foo
- diff: svn diff -r123 foo
- import: svn import svn://foo -m “import msg”

Basic Subversion Commands (2/2)

- info: svn info
- list (ls): svn ls svn://foo
- log: svn log foo
- mkdir: svn mkdir foo
- move (mv): svn mv svn://foo svn://bar
- resolved: svn resolved foo
- status: svn status -u
- update (up): svn up

Debugging in NetBeans

Configurable Debugger: In the Options dialog, you can configure breaking/suspending behavior, you can specify Variable Formatters, and skip methods and packages using Step Filters.

Debugging Window: The Debugging window integrates the Sessions, Threads and Call Stack views.

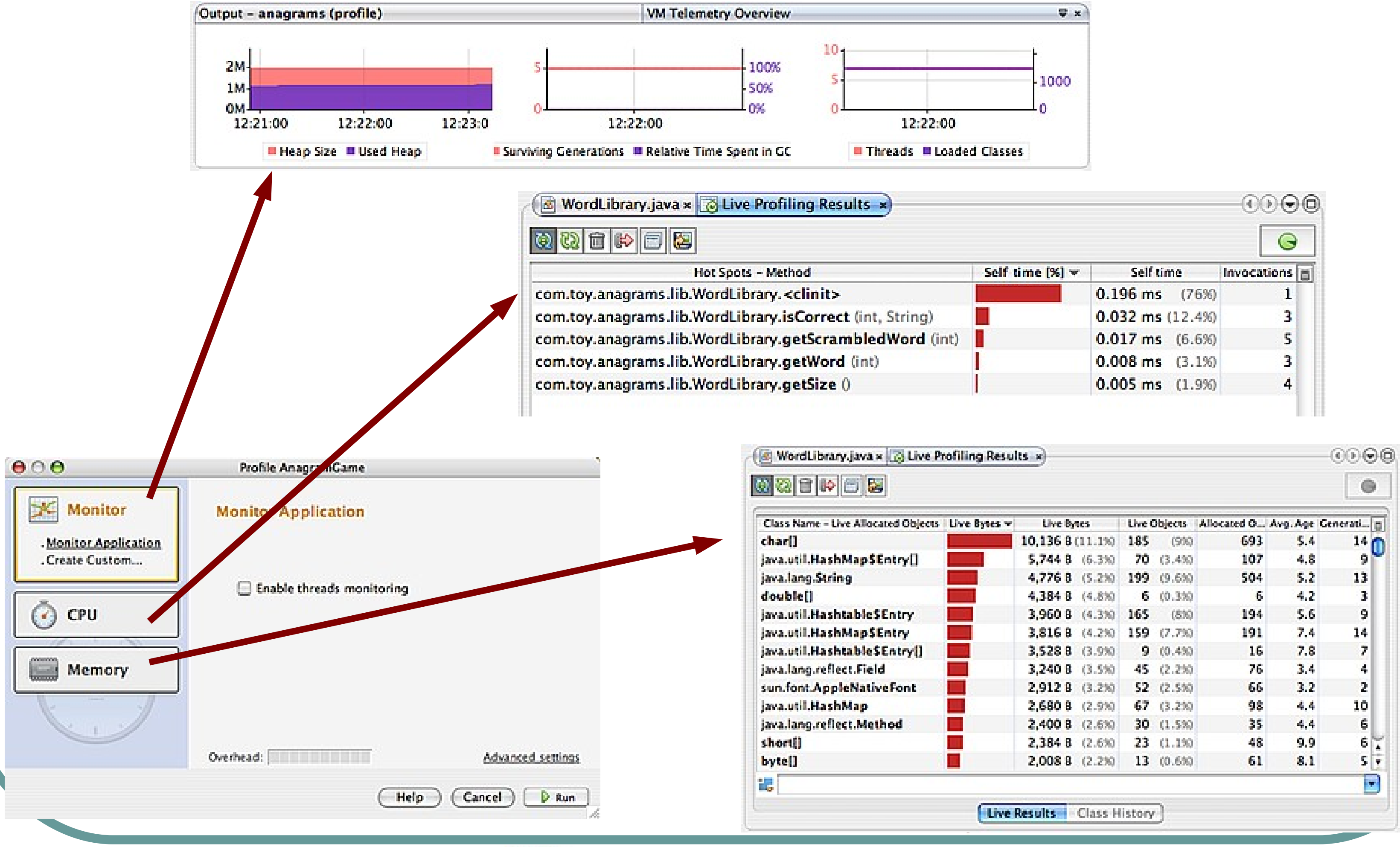
Each sessions is broken down in its **list of threads**, and you can expand each suspended thread to its call stack, etc. You can resume/suspend threads with one click on the play/pause buttons.

Configurable Breakpoints: In addition to the **standard line and method breakpoints**, the NetBeans debugger provides advanced **Class, Thread, Exception, and Variable breakpoints**. Configure these custom breakpoints to be triggered by conditions and events such as uncaught exceptions, class loading, or variable access.

Expression Evaluation: Evaluate Java-syntax expressions assigned to watches and conditional breakpoints "live" while stepping through your code. Moving the pointer over the variable and the current value is evaluated and displayed in a tool tip.

Expression Stepping: You can easily step over individual expressions within a statement. The debugger will display the return value from each expression. The Step Into action (F7) lets you select the method call to step into if there is more than one possibility at the current line.

Profiling



☐ Enable threads monitoring

Hot Spots – Method	Self time [%]	Self time	Invocations
com.toy.anagrams.lib.WordLibrary.<clinit>		0.196 ms (76%)	1
com.toy.anagrams.lib.WordLibrary.isCorrect (int, String)		0.032 ms (12.4%)	3
com.toy.anagrams.lib.WordLibrary.getScrambledWord (int)		0.017 ms (6.6%)	5
com.toy.anagrams.lib.WordLibrary.getWord (int)		0.008 ms (3.1%)	3
com.toy.anagrams.lib.WordLibrary.getSize ()		0.005 ms (1.9%)	4

Class Name – Live Allocated Objects	Live Bytes	Live Bytes	Live Objects	Allocated O...	Avg. Age	Generati...
char[]		10,136 B (11.1%)	185 (9%)	693	5.4	14
java.util.HashMap$Entry[]		5,744 B (6.3%)	70 (3.4%)	107	4.8	9
java.lang.String		4,776 B (5.2%)	199 (9.6%)	504	5.2	13
double[]		4,384 B (4.8%)	6 (0.3%)	6	4.2	3
java.util.Hashtable$Entry		3,960 B (4.3%)	165 (8%)	194	5.6	9
java.util.HashMap$Entry		3,816 B (4.2%)	159 (7.7%)	191	7.4	14
java.util.Hashtable$Entry[]		3,528 B (3.9%)	9 (0.4%)	16	7.8	7
java.lang.reflect.Field		3,240 B (3.5%)	45 (2.2%)	76	3.4	4
sun.font.AppleNativeFont		2,912 B (3.2%)	52 (2.5%)	66	3.2	2
java.util.HashMap		2,680 B (2.9%)	67 (3.2%)	98	4.4	10
java.lang.reflect.Method		2,400 B (2.6%)	30 (1.5%)	35	4.4	6
short[]		2,384 B (2.6%)	23 (1.1%)	48	9.9	6
byte[]		2,008 B (2.2%)	13 (0.6%)	61	8.1	5

Junit

- “I am not a grat programmer, I am just a good programmer with great habits”

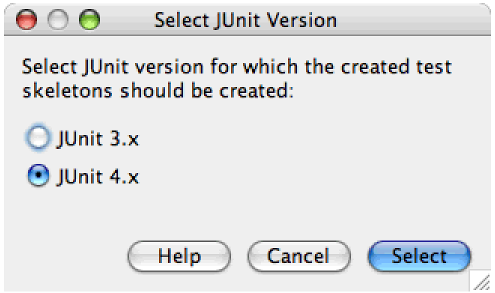
Kent Beck

- Possiamo creare test per ogni calsse implementando il main e stampando stringhe a terminale
- Junit è un piccolo framework che consente di creare e gestire test:
  - Semplici
  - Automatici
  - Componibili
  - Isolati
  - Orientati agli oggetti

Generare test in NetBeans

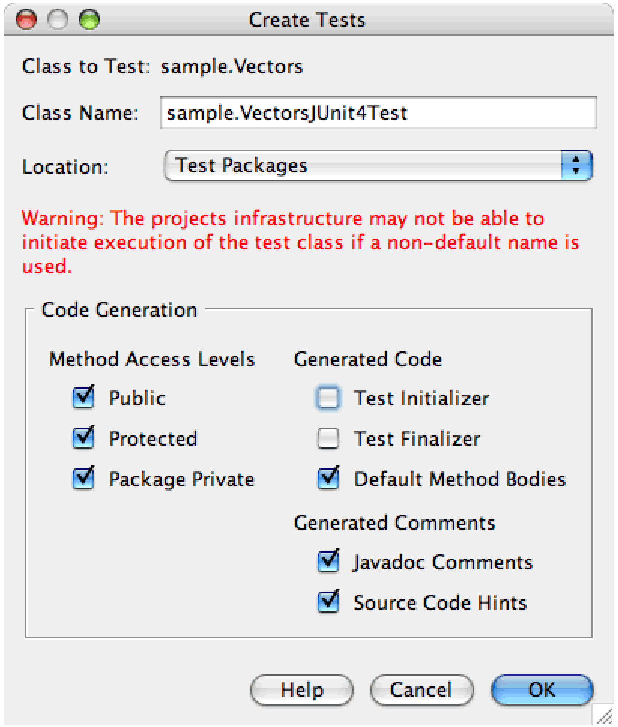
- Click destro su una classe e

  Tools > Create Junit Tests



Select JUnit version for which the created test skeletons should be created:

○ JUnit 3.x
● JUnit 4.x



Class to Test: sample.Vectors

Class Name: sample.VectorsJUnit4Test

Location: Test Packages

Warning: The projects infrastructure may not be able to initiate execution of the test class if a non-default name is used.

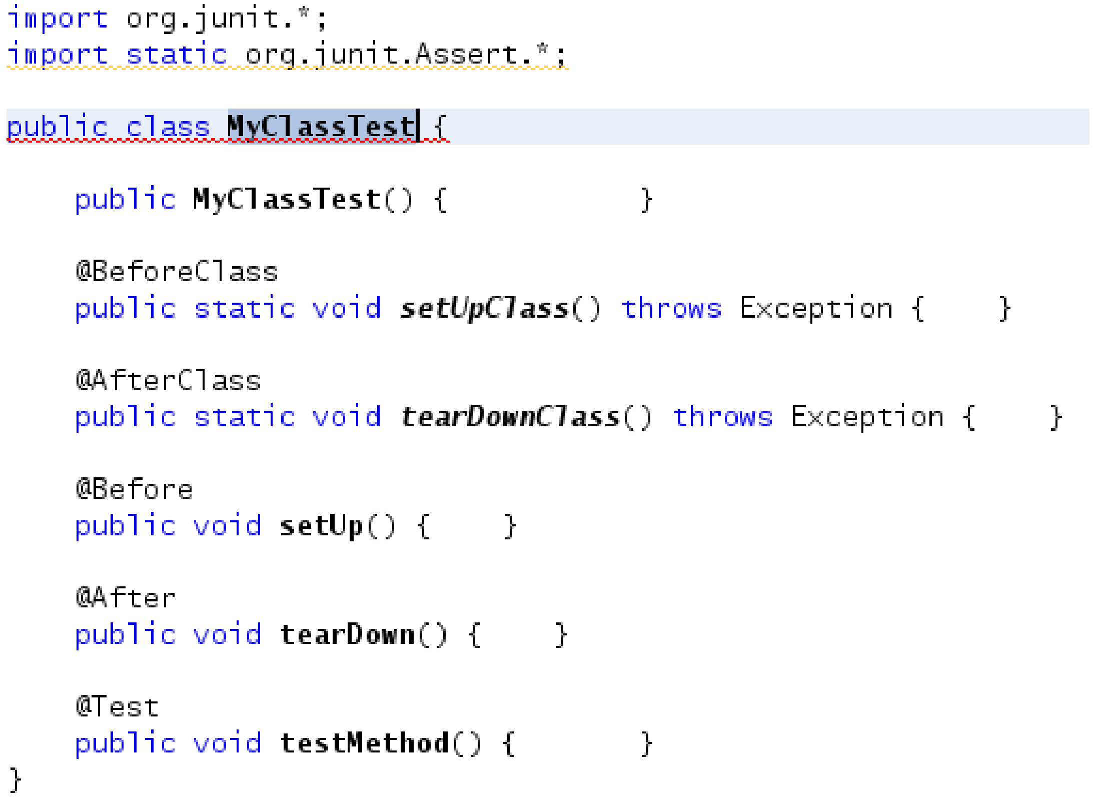
Code Generation

Method Access Levels
☑ Public
☑ Protected
☑ Package Private

Generated Code
☐ Test Initializer
☐ Test Finalizer
☑ Default Method Bodies

Generated Comments
☑ Javadoc Comments
☑ Source Code Hints

Struttura di un test



```
import org.junit.*;
import static org.junit.Assert.*;

public class MyClassTest {

    public MyClassTest() {            }

    @BeforeClass
    public static void setUpClass() throws Exception {     }

    @AfterClass
    public static void tearDownClass() throws Exception {     }

    @Before
    public void setUp() {     }

    @After
    public void tearDown() {     }

    @Test
    public void testMethod() {        }
}
```

- Per lanciare un test: click destro su esso e selezionare “Test File”

Assert

- Test definiti tramite l’uso della famiglia di AssertXxx()
  - AssertTrue() / AssertFalse()
  - AssertNull() / AssertNotNull()
  - AssertEquals() / AssertArrayEquals()
  - AssertSame() / AssertNotSame()
  - Fail()
- Tutti i dettagli nelle API di JUnit

Opzioni per i Test

- Timeout

  `@Test(timeout=1000)`

- Eccezione Attesa

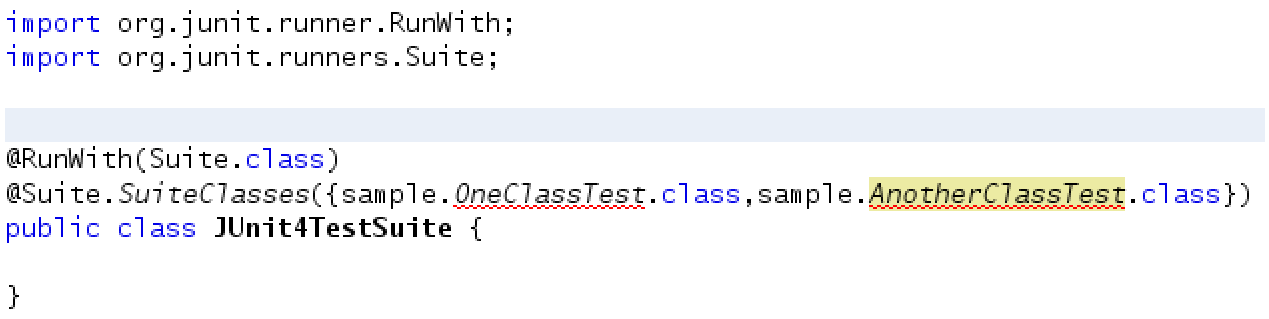
  `@Test(expected=IOException.class)`

- Disabilitare un test

  `@Ignore @Test`

Test Suite in NetBeans

- Click destro sul progetto, new Junit test suite
- Specificarere le classi di test nell'annotazione @Suite.SuiteClasses({...})



```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;


@RunWith(Suite.class)
@Suite.SuiteClasses({sample.OneClassTest.class,sample.AnotherClassTest.class})
public class JUnit4TestSuite {

}
```

Esercizio

- Creare una classe MyCounter che implementi e gestisca un oggetto di tipo contatore, con reset, incremento e getCount (non statici!)
- Creare un test JUnit per un oggetto di tale classe
- Creare una sottoclasse MyNewCounter, che estende il comportamento (inizializzazione a valore diverso da 1, decremento)
- Creare un test Junit per questa sottoclasse, estendendo il precedente
- Gestire e testare anche il lancio di eccezioni (Es. inizializzazione negativa, decremento sotto 0)