

```

<path id="classpath.test">
  <pathelement location="lib/junit-4.13.1.jar"/>
  <pathelement location="lib/hamcrest-core-1.3.jar"/>
  <pathelement location="../Fantaisie-Impromptu [Act3]/build"/>
</path>

<target name="compile.java" depends="clean, init">
  <javac srcdir="../Fantaisie-Impromptu [Act3]/src/classes" destdir="../Fantaisie-Impromptu [Act3]/build"
    includeantruntime="false"/>
</target>

<target name="compile.test" depends="compile.java">
  <javac srcdir="../Fantaisie-Impromptu [Act3]/src/test" destdir="../Fantaisie-Impromptu [Act3]/build/test"
    includeantruntime="false">
    <classpath refid="classpath.test"/>
  </javac>
</target>

<target name="compile" depends="compile.java, compile.test, jar_comp"/>

```

compile.java:  
[javac] Compiling 2 source files to C:\Users\Hans\Desktop\Fantaisie-Impromptu [act4]\build

compile.tests:  
[javac] Compiling 1 source file to C:\Users\Hans\Desktop\Fantaisie-Impromptu [act4]\build\tests

Task name:	Compile task
Description:	This task will compile the needed classes for testing, which are Fraction.java and MixedFraction.java, including the test class FractionTest.java.

```

<target name="tests" depends="compile">
  <jacoco:coverage destfile="lib/jacoco.exec">
    <junit printsummary="yes" haltonfailure="yes" haltonerror="yes" fork="yes">
      <formatter type="plain"/>
      <classpath>
        <path refid="classpath.tests"/>
        <pathelement location="build/classes"/>
        <pathelement location="build/tests"/>
      </classpath>
      <formatter type="plain"/>
      <batchtest fork="yes" todir="build/junit-reports">
        <fileset dir="src/tests">
          <include name="FractionTests.java"/>
        </fileset>
      </batchtest>
    </junit>
  </jacoco:coverage>
</target>

```

```

test:
[jacoco:coverage] Enhancing junit with coverage
[junit] Running FractionTest
[junit] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.037 sec

```

Task name:	JUnit testing task
Description:	This task is responsible for running the compiled test class using JUnit and Hamcrest core jar files, and it will write the results in the JUnit-reports directory. Lastly, the code coverage used for this task is Jacoco, which will generate code coverage in the next task.

```

<target name="coverage-report" depends="tests">
  <jacoco:report>
    <executiondata>
      <file file="lib/jacoco.exec"/>
    </executiondata>
    <structure name="Jacoco Coverage Report">
      <classfiles>
        <fileset dir="build/tests">
          <include name="*.class" />
        </fileset>
        <fileset dir="build/classes">
          <include name="Fraction.class" />
        </fileset>
      </classfiles>
      <sourcefiles>
        <fileset dir="src/tests">
          <include name="*.java"/>
        </fileset>
        <fileset dir="src/main">
          <include name="Fraction.java" />
        </fileset>
      </sourcefiles>
    </structure>

    <html destdir="build/coverage-reports" />
    <xml destfile="build/coverage-reports/tests-coverage-report.xml" />
    <csv destfile="build/coverage-reports/tests-coverage-report.csv" />
  </jacoco:report>
</target>

```

coverage-report:

[jacoco:report] Loading execution data file C:\Users\Hans\Desktop\Fantasia-Improptu [Act3]\lib\jacoco.exec

[jacoco:report] Writing bundle 'Jacoco Coverage Report' with 2 classes

[jacoco:report] To enable source code annotation class files for bundle 'Jacoco Coverage Report' have to be compiled with debug information.





















Task name:	Coverage report task
Description:	This task is responsible for generating coverage reports taken from the previous testing task, and the outputs for the report are in a html, xml, and csv format.

## Jacoco FractionTest class Report

### Fraction

Class files must be compiled with debug information to show line coverage.

Class files must be compiled with debug information to link with source files.

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Methods
● read(String)		0%		0%	6	6	1	1
● Fraction(int)		0%		n/a	1	1	1	1
● toDouble()		0%		n/a	1	1	1	1
● add(Fraction)		100%		n/a	0	1	0	1
● subtract(Fraction)		100%		n/a	0	1	0	1
● reduce()		100%		n/a	0	1	0	1
● multiplyBy(Fraction)		100%		n/a	0	1	0	1
● divideBy(Fraction)		100%		n/a	0	1	0	1
● toString()		100%		100%	0	2	0	1
● toMixedFraction()		100%		n/a	0	1	0	1
● computeGCD(int, int)		100%		100%	0	2	0	1
● Fraction()		100%		n/a	0	1	0	1
● Fraction(int, int)		100%		n/a	0	1	0	1
● setNumerator(int)		100%		n/a	0	1	0	1
● setDenominator(int)		100%		n/a	0	1	0	1
● getNumerator()		100%		n/a	0	1	0	1
● getDenominator()		100%		n/a	0	1	0	1
Total	72 of 291	75%	10 of 14	28%	8	24	3	17










#### Description:

The fraction class in the coverage report shows the methods used during testing. The html file shows the percentage of branches and instructions missed or executed while the test was running. The test aims to utilize fraction operations with proper and improper fractions, accessor, and mutator methods, which is why method read and to double, and constructor for a whole number fraction was not used.

## FractionTest

Class files must be compiled with debug information to show line coverage.

Class files must be compiled with debug information to [link with source files](#).

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Methods
mutators()		100%		n/a	0	1	0	1
add()		100%		n/a	0	1	0	1
subtract()		100%		n/a	0	1	0	1
multiply()		100%		n/a	0	1	0	1
divide()		100%		n/a	0	1	0	1
accessors()		100%		n/a	0	1	0	1
FractionTest()		100%		n/a	0	1	0	1
reduce()		100%		n/a	0	1	0	1
convertFraction()		100%		n/a	0	1	0	1
Total	0 of 245	100%	0 of 0	n/a	0	9	0	9

Description:

The FractionTest class basically shows that all the test methods under the test suite are executed.