ADVANCED JAVA PROGRAMMING AJP-P1-2012-2013

BEFORE YOU START

Please take some time to familiarise yourself with the NetBeans IDE. Please work through the following online tutorials:

The Netbeans IDE Java quick start tutorial
http://netbeans.org/kb/docs/java/quickstart.html

Developing General Java Applications
http://netbeans.org/kb/docs/java/javase-intro.html

Code Assistance in the NetBeans IDE
http://netbeans.org/kb/docs/java/editor-codereference.html

Keyboard shortcuts
http://netbeans.org/projects/www/downloads/download/shortcuts.pdf

CONFIGURE CHECKSTYLE

CheckStyle has been installed on the lab machines. All you need to do is configure it.

Go to *EAT*. Download the config file from AJP → Learning Materials → Labs to your home folder.

Open up *Netbeans*. Go to Tools → Options → Misc. → CheckStyle

Click in the field labelled *Config File*. Browse to the config file you downloaded. Press OK.

You have configured *CheckStyle*. You only have to do this once.

INTRODUCTION

Here is a revision problem to refresh your coding skills. It is called *PizzaSorter*.

A restaurant cooks three types of pizzas (9 inch, 12 inch and 15 inch).

Each pizza can have two different toppings (e.g. cheese, tomato, sausage etc.).

Twelve inch pizza can have a stuffed crust (either cheese or spicy stuffing).

Fifteen inch pizzas can also have a stuffed crust and are available in deep pan or thin pan.

Your application will read a text file containing a list of pizzas ordered by customers.

The application will process these orders and output sorted groups on demand.

SET-UP

Import the Netbeans project folder named 'AJP-P1-2012-2013-STUDENT'. You should have the following files:

Pizza.java

This class describes a generic pizza in abstract terms.

SmallPizza.java

This is a concrete implementation of Pizza. You should not alter this file.

PIzzaSorter.java

This class contains three blank methods which you will implement:

- parseFile(File file)
- listAllPizzas(String fileName)
- filterPizzas(String fileName, int radius)

pizza.txt.

This is a read only file containing several lines of data. Do not edit this file. Each line of data represents one Pizza.

Each line starts with an integer argument. This integer represents the radius (in inches) of the pizza being described.

After a single space, there will be another String argument. This is the 1st pizza topping.

After a single space, there will be another String argument. This is the 2nd pizza topping.

If the pizza is a twelve inch or fifteen inch pizza, there will be another String argument. This describes the type of stuffed crust that was requested.

If the pizza is a fifteen inch pizza, there will be another String argument. This describes the pizza thickness (deep pan or thin pan).

An example of the type of data found in pizza.txt is shown below:

```
9 cheese tomato
12 chicken mushroom cheese
15 sausage bacon cheese deep-pan
```

This file will be in the **root** of the project folder.

wrong-num. txt

This is a read only file containing several lines of data. Do not edit this file. Note that one line of data contains invalid data – 4 inch pizza! This file will also be in the **root** of the project folder.

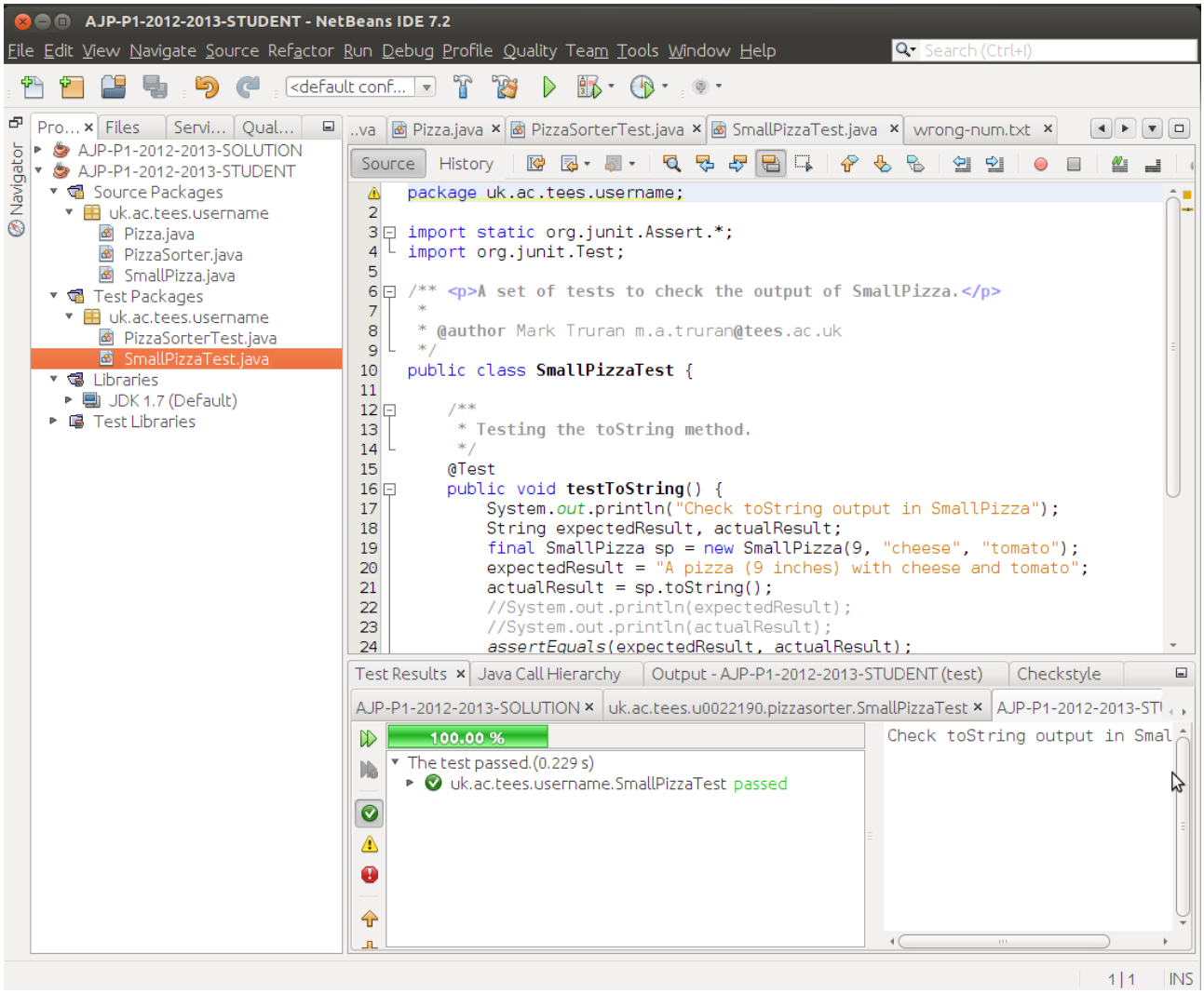
Test Cases

Your project also contains various unit tests, which you can find in the folder marked 'Test Packages'.

Before you start, under *Source Packages*, replace uk.ac.tees.**username** with your actual username.

INSTRUCTIONS FOR BRONZE AWARD

1. Let's try a unit test out!. Expand the folder marked 'Test Packages', highlight the file named *SmallPizzaTest.java*. Right click on the file. Select *Test File*. You should see something like this:



This tells us that the *toString*() method of the *SmallPizza* class is behaving as expected i.e. when a *SmallPizza* object is built with cheese and tomato toppings, and the *toString*() method is called, the return value is:

```
A pizza (9 inches) with cheese and tomato
```

Make some changes to *TestSmallPIzza* so that it now reads

```
System.out.println("Check toString output in SmallPizza");
String expectedResult, actualResult;
final SmallPizza sp = new SmallPizza(9, "cheese", "tomato");
expectedResult = "A delicious pizza (9 inches) with cheese and tomato";
actualResult = sp.toString();
System.out.println(expectedResult);
System.out.println(actualResult);
assertEquals(expectedResult, actualResult);
```

Re-run the test. It should now fail, because the expected output and the actual output do not match. Fix the test case, and re-run using the 'rerun' button on the left hand side of the *Test Results* tab. It looks like two overlapping play buttons.

Test cases give us an way to check the output of a method automatically. Hooray!

2. Return to *Source Packages*. Create a new class called *MediumPizza.java*. This class should extend the *Pizza* class. Use *SmallPizza.java* as a guide. You should provide a constructor with four parameters (radius, topping, topping, crust). Your constructor should call the superconstructor. You should override the *toString*() method so that it produces a textual description in the following format:

```
A pizza (12 inches) with ham and pineapple and a cheese crust
```

3. Create a new class called *LargePizza.java*. This class should extend the *Pizza* class. Use *SmallPizza.java* as a guide. You should provide a constructor with five parameters (radius, topping, topping, crust, thickness). Your constructor should call the superconstructor. You should override the *toString*() method so that it produces a textual description in the following format:

```
A pizza (15 inches) with sausage and bacon and a cheese crust, deep-pan
```

4. Under *Source Packages*, open the file named *PizzaSorter*. Now fill in the method named *parseFile*() so

- It creates a *Scanner* object. This *Scanner* should be initialised with the *File* object the *parseFile*() method is passed when invoked.
- Read each line of text in the file
- If the line of text describes a nine inch pizza, instance the *SmallPizza* class and add this object to the *ArrayList*.
- If the line of text describes a twelve inch pizza, instance the *MediumPizza* class and add this object to the *ArrayList*.
- If the line of text describes a fifteen inch pizza, instance the *LargePizza* class and add the object to the *ArrayList*.
- Return the *ArrayList*

When you think your code is correct, open up the file named *PizzaSorterTest.java* in the *Test Packages*. Above the method named *testParseValidFile*(), remove the annotation that reads **@Ignore**. It should now read:

```
@Test
public void testParseValidFile() {


  System.out.println("Parsing valid file");
  final PizzaSorter ps = new PizzaSorter();
  final ArrayList<Pizza> listOfPizzas = ps.parseFile(new File("pizza.txt"));
  final int expectedResult = 9;
  final int actualResult = listOfPizzas.size();
  assertEquals(expectedResult, actualResult);
}
```

Right click on *PizzaSorterTest.java.* Select *'Test File'* If your code is correct, the newly added test will pass. If not, you have an error somewhere. You should have parsed 9 Pizza objects from the file. Make sure this is working before you move on.

5. In *Source Packages,* modify *parseFile*() so that your method returns NULL if either of the following things occurs:

1. The *File* object is not found in the root folder so that a *FileNotFoundException* is thrown.
2. The method is passed a valid *File* object but the file contains invalid data (incorrect pizza radius).

When you think your code is correct, open up the file named *PizzaSorterTest.java* in the *Test Packages*. Above the method named *testParseInvalidFileName*(), remove the annotation that reads **@Ignore**. Above the method named *testParseInvalidData*(), remove the annotation that reads **@Ignore**. Right click on *PizzaSorterTest.java.* Select *Test File.* If your code is correct, the newly added tests will pass.

6. Under *Source Packages*, open the file named *PizzaSorter*. Now fill in the method named *listAllPizzas*() so that it

- Creates an *ArrayList* capable of storing *Pizza* objects
- Calls the *parseFile*() method, passing it a *File* object. You should initialise the File object using the *String* object the *listAllPizzas*() method is passed when invoked.
- Initialise the *ArrayList* using the return value of the call to the *parseFile*() method.
- Create a new *String* object called *list*
- Iterate through the *ArrayList* of *Pizza* objects calling the *toString*() method of each object, adding the return value of the method call plus a newline character to the *String* called *list*
- Return *list*.

When you think your code is correct, open up the file named *PizzaSorterTest.java* in the *Test Packages*. Above the method named testListAllPizzas(), remove the annotation that reads **@Ignore**. Right click on *PizzaSorterTest.java.* Select *Test File.* If your code is correct, the newly added test will pass.

7. Under *Source Packages*, open the file named *PizzaSorter*. Now fill in the method named *filterPizzas*() so that it

- Creates an *ArrayList* capable of storing *Pizza* objects
- Calls the *parseFile*() method, passing it a *File* object. You should initialise the *File* object using the *String* object the *listAllPizzas*() method is passed when invoked.
- Initialise the *ArrayList* using the return value of the call to the *parseFile*() method.
- Create a new *String* object called *list*
- Iterate through the *ArrayList* of *Pizza* objects. If the pizza object has the same radius as the argument passed to the filterPizzas() method, call the *toString*() method of the object, then add the return value of that method call plus a newline character to the *String* called *list.*

When you think your code is correct, open up the file named *PizzaSorterTest.java* in the *Test Packages*. Above the methods named *testFilterPizzas9()*, *testFilterPizzas12()* and *testFilterPizzas15(),* remove the annotation that reads **@Ignore**. Right click on *PizzaSorterTest.java*. Select *Test File* If your code is correct, the newly added test will pass.

8. Right click on the project name (AJP-P1-2012-2013) and select *Test*. This runs <u>all</u> unit tests in the project. If they all pass, then you have completed the basic task (Bronze award).

INSTRUCTIONS FOR SILVER AWARD

If you would like to try for a SILVER award, make sure that your *CheckStyle* report has no errors.

INSTRUCTIONS FOR GOLD AWARD

If you would like to try for a GOLD award, you need to complete the following advanced tasks.

1. Using *SmallPizzaTest* as an example, write unit tests for the *toString*() methods of the *MediumPizza* and *LargePizza* classes.
2. In steps (6) and (7) above, we repeatedly concatenate a *String* object. This is inefficient. Find a better solution.
3. At the moment we can create pizzas with null string toppings. Fix this.