**CSE 1325 – Spring 2013**

**Object-Oriented and Event-driven Programming in Java**

**Lab 1**

Assigned: Thursday Jan 17th 2013

Due: Tuesday 11:59 pm Jan 29th 2013

**Objective:**

The goal of this lab is to get you started programming in Java quickly by learning the basics of Java’s input and output methods, the popular Netbeans Integrated Development Environment (IDE), as well as some basic Object Oriented Programming (OOP). In this project, you will create a class and test program.

**Description:**

The C program shown below is an example of the procedural programming approach to finding the distance between two points. In this project you will be implementing this functionality in Java in an Object Oriented approach.

#include<stdio.h>

struct coordinates**{**

int x**;**

int y**;**

**}**points**[**2**];**

int main**(**unsigned int argc**,** char **\*** argv**[]){**

ask\_user\_for\_points**(**points**);**

distance\_between\_points**(**points**[**0**],** points**[**1**]);**

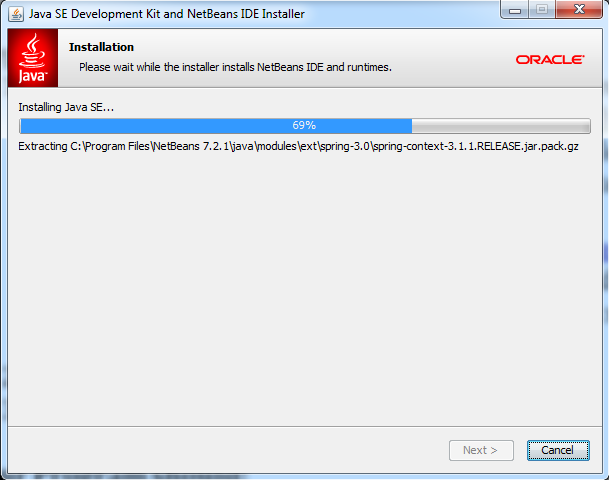
**return** 0**;**

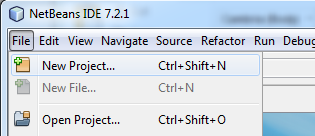
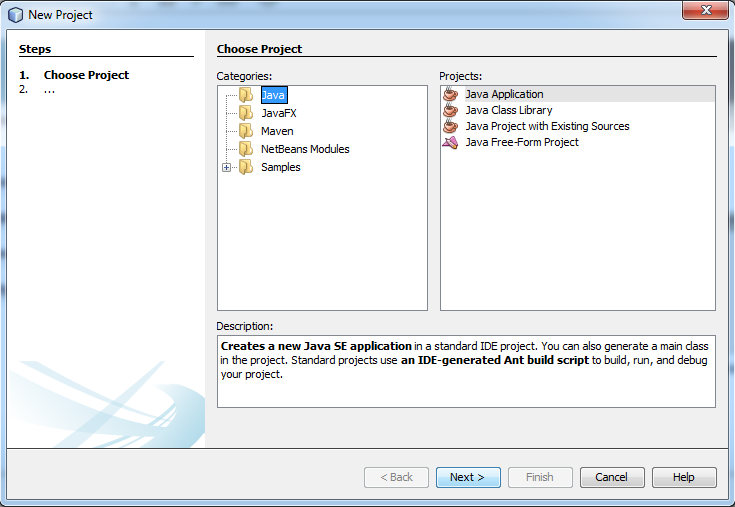
**}**

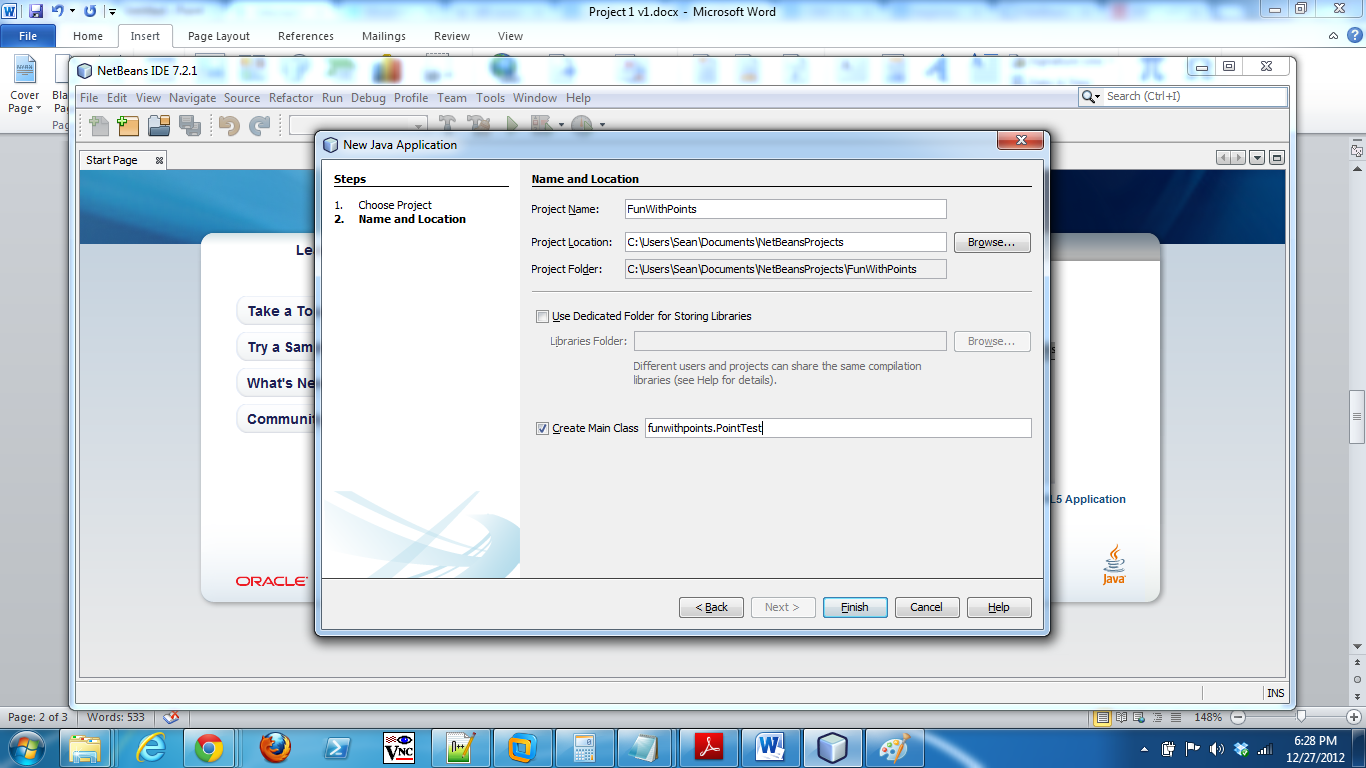
In the C programming language it is possible to write all your code in one file, however this is considered bad programming practice for large projects and Java enforces proper design by constraining each class to a particular file. In this project you will have two files for two classes:

* You will have a file called, ‘Point.java’
* You will have another file called, ‘PointTest.java’

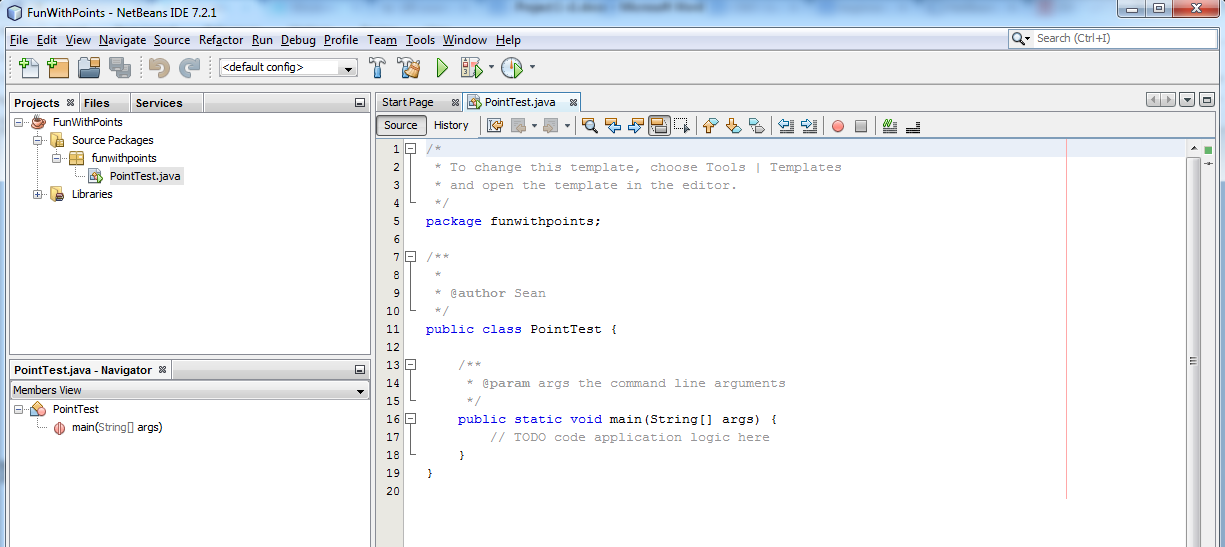
Follow the steps below to create the two classes.

**Steps to making your Development Environment:**

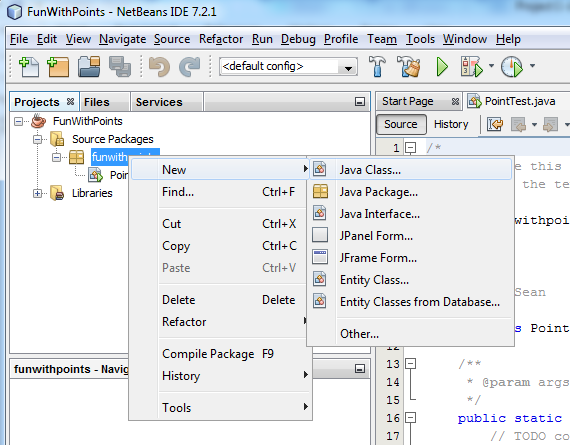
1. Download and install the latest Oracle Official Java Software Development Kit Standard Edition (Java SDK SE), JUnit, and the Netbeans IDE. I suggest you do this via the bundle referred to as The Java Development Kit (JDK). Download the architecture specific bundle for your computer’s here: <http://www.oracle.com/technetwork/java/javase/downloads/jdk-7-netbeans-download-432126.html>. When in doubt of your computer’s architecture it’s probably ‘Windows x86’. If you have any questions email me at [seanp@uta.edu](mailto:seanp@uta.edu)
2. Start Netbeans:
3. Create a new project.
4. Under the 'Categories' make sure 'Java' is selected and under, 'Projects:' make sure, 'Java Application' is selected, then click 'Next >'.
5. Use the Project name, ‘FunWithPoints’ and make sure ‘Create Main Class’ has FunWithPoints.PointTest’



1. Click ‘Finish’ and you should see a page exactly as the one below:



1. Create the Point class by right-clicking on the ‘funwithpoints’ package icon in the Projects Menu. Then click ‘New’ then, ‘Java Class’.

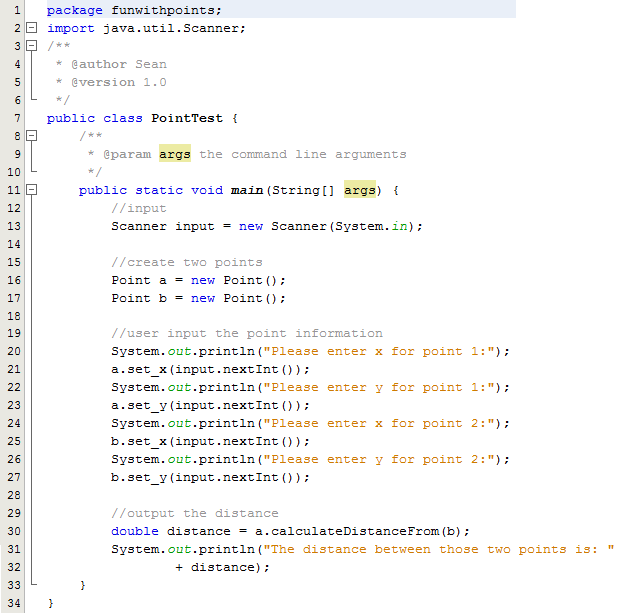


1. Type ‘Point’ into ‘Class Name’ then click ‘Finish’



**How to start programming**

* Along with this document you should have downloaded a Point.java file to use for this project. If NetBeans sees the file, then you can use it.
* If you didn't download the Point.java file (or can't see it), you can copy and paste the given code into your current Point.java file.
* Take a look at the code in the Point.java file and the corresponding comments. You should see that the Point class has:
* the private variables:
  + int x
  + int y
* the public methods:
  + void set\_x(int userGivenX)
  + void set\_y(int userGivenY)
  + int get\_x()
  + int get\_y()
  + float calculateDistanceFrom(Point somePoint)
* and more explanation further down below on this sheet
* In the PointTest.java file you will only have one method which should be auto generated by the steps above. It is:
  + public static void main(String[] args)
* Now insert this code into the ‘main’ method in the file PointTest.java:



**Compile your Code**

To compile and run your code locally in Netbeans press the  key.

Since you will also be required to compile this code on a unix-style environment, then you need to be able to compile java source code from the command line. As a minimum requirement you must do this on omega.uta.edu. To access a shell on omega.uta.edu you can use UTA’s [SSH client](http://www.uta.edu/oit/cs/software/ssh/ssh-secure-shell-for-workstations-3/index.php). If you need help accessing omega.uta.edu see OIT’s help site [here](http://www.uta.edu/oit/cs/web/OmegaUtaEdu-FAQ.php).

You'll need to put your files on omega.uta.edu in your account. We suggest that you create a directory for your Java classes and put all the files for your Java code into that directory.

To compile the source code in Linux/Unix (and Windows) navigate to the source code directory where all your .java files are stored (usually with the ‘cd’ command) the type in ‘javac \*.java’. To run your code, type the name of file with your main function without the ‘.java’ extension. Example, ‘java PointTest’ will execute the code above in the main method. If you have problems with these commands then you probably need to add the Java SDK directory to your environmental path.

In order to demonstrate that your Java classes compile and run on omega, you should create a script file showing the compile step and running the program. When you turn in your Java class code you should also submit the script file as well.

Make sure that your classes follow the Java conventions given below and that your lab submission follows the instructions given below.

In future labs you will need to implement more Java classes. They will be described in terms of members and functions as shown above in the Point and TestPoint descriptions above.

**If your program runs, then it should:**

1. Take in two integer points from the user from the command line
2. Display the distance between those two points as seen below.

Please enter x for point 1:

1

Please enter y for point 1:

1

Please enter x for point 2:

2

Please enter y for point 2:

2

The distance between those two points is: 1.4142135623730951

Another example would be:

Please enter x for point 1:

0

Please enter y for point 1:

0

Please enter x for point 2:

0

Please enter y for point 2:

0

The distance between those two points is: 0.0

You should run and test your code with at least three additional value pairs for point 1 and point 2.

**Standard Java conventions and coding style:**

Be sure to observe standard Java naming conventions and style. These will be observed across all projects for this course, hence it is necessary that you understand and follow them correctly. They include:

* Class names begin with an upper-case letter, as do any subsequent words in the class name.
* Method names begin with a lower-case letter, and any subsequent words in the method name begin with an upper-case letter. This is called "camelCase".
* Class, instance and local variables begin with a lower-case letter, and any subsequent words in the name of that variable begin with an upper-case letter.
* Each set of opening/closing curly braces must be in the same column. All user prompts must be clear and understandable
* Give meaningful names for classes, methods, and variables even if they seem to be long. The point is that the names should be easy to understand for a person looking at your code
* All static final variables will be upper case identifiers

In addition, ensure that your code is properly documented in terms of comments and other forms of documentation wherever necessary.

**What and How to Submit**

For this project please submit using the Blackboard system at elearn.uta.edu. Please submit a .zip file that contains:

* Each .java file used in development with the appropriate name for the included class
* A script file of the session on omega
* A short description of the method calculateDistanceFrom from the Point class including what the inputs are, what calculation is done, and how the result is returned from the method
* A development notes file (standard text or word file) that describes at least one error that you found while developing your code, how you solved it, and the total amount of time you spent on this project in terms of hours.

The .zip file containing your project must have the name format of: **proj1\_1000number\_firstname\_lastname**

**Additional explanation of the Point code:**

* Data type Casting is the same as in C.
* To get an integer from the command line, i.e. input typed in by the user, you will need the scanner library so you will need to import this library:

import java.util.Scanner;

Then the Scanner Object will become available:

Scanner input = new Scanner(System.in);

input.nextInt();

* To print anything out to the command line you will need to reference System.out.println:

System.out.println(“Hello World”);

* The Java Math library is extremely useful and does NOT have to be imported, it can be simply accessed by:

int x = 3;

x = Math.pow(x,2);

x = Math.sqrt(x);

**A Few Helpful Tips:**

* Netbeans by default recompiles your class every time you save your file. If you want to force everything to be recompiled, press the ‘clean and build’ button.
* Ctrl + space in most IDE’s will show a list of available members (or variables) and methods (or functions).

**Grading Rubric:**

**Code: (50%)**

Program header and function headers for all functions (10 points)

Comments (line comments and block comments) (10 points)

Modularity (division of the problem into small parts) (10 points)

Style (indentation, consistency, meaningful identifiers, lateral separation of code from line comments, etc.) (10 points)

Correct declaration and manipulation of the PointTest code (5 points)

Proper implementation of data input (5 points)

**Output: (50%)**

User clearly understands what is being requested for input (5 points)

Calculate distance task performs correctly (5 points)

Set functions handle inputs correctly (5 points)

Get functions provide values safely (5 points)

Output gives clear information to explain the values to the user (5 points)

Output contains all the given test data and at least three additional data sets, i.e at least three other pairs of points (25 points)

**Grading Deductions:**

Use of global variables will result in an overall grade of 0 (zero) [-100 deduction]

Use of unstructured control elements, like the C break or continue command, will result in an overall grade of 0 (zero) [-100 deduction]

Labs which have errors in them and do not terminate normally will receive an overall grade of 0 (zero) [-100 deduction]

Late submission of softcopy of code and/or script file or other documents to Blackboard will result in an overall grade of 0 (zero) UNLESS student has obtained prior instructor approval [-100 deduction]

Use of Java language elements not yet discussed in class by the lab due date will result in potential deduction of points - discuss with instructor before using.

Miscellaneous: If you have questions, e-mail Dr. T ([tiernan@uta.edu](mailto:tiernan@uta.edu)) and the TA (see website for contact info)