CSE332: Data Abstractions

Section 1

Hye In Kim

Spring 2014

Section Agenda

- Introduction
- Generics
- Project 1: Sound Blaster!
- Bugs & Testing
- HW1 Tips
- Eclipse Tutorial

Introduction

Introduction - Me

- Hye In (Han) Kim
- From Korea
- 5th year Master's student
- BS in Biology & CSE at UW
- Teaching section, Grade assignments

- Office hour: F 01:30 ~ 02:30, CSE 220
- Email: kainby87@uw.edu

Introduction - You

- Name
- Year
- Home Town
- Interesting Fact about yourself

Generics

Generics

- **What is generics?**

  - Technique of writing class/Interface without specifying type of data it uses

  - Idea: class/interface can have type parameter
    Usually denoted as T or E

Generics

- **Want a Bag class to store Items**

```
public class Bag {                    // Stores String
        private String item;
        public void setItem(String x) {  item = x;  }
        public String getItem( )  {  return item;  }
}

public class Bag {                    // Stores Book
        private Book item;
        public void setItem(Book x) {  item = x;  }
        public Book getItem( )  {  return item;  }
    }
```

- **Problem**? Don't want to make Bag class for all kind of fields.

Generics

- **Want a Bag class to store Items**

```
public class Bag<E> {
        private E item;
        public void setItem(E x) {  item = x;  }
        public E getItem( )  {  return item;  }
}

Bag<String> b = new Bag<String>();
b.setItem("How about this?”);
String contents = b.getItem();
```

- Can we accomplish same effect without using generics?

Generics

- **Want a Bag class to store Items**

  - Pre Java 5: Objects

```
public class Bag {
        private Object item;

        public void setItem(Object x ) {   item = x;   }
        public Object getItem() {  return item;  }
}

Bag b = new Bag();
b.setItem("How about that?”);
String contents = (String) b.getItem();
```

Generics

- **Why generics?**

```
Bag b = new Bag();                                  // Object Bag
b.setItem( "How about that?" );
String contents = (String) b.getItem();             // Ok
double contents = (double) b.getItem();             // Error (Runtime)

Bag<String> b = new Bag<String>();                  // Generic Bag
b.setItem( "How about that?" );
String contents = b.getItem();                      // Ok
double contents = b.getItem();                      // Error (Compile time)
```

Generics

- **Why generics?** Type Safe Containers

  - Main advantage: compile-time type checking

  - Generics: Ensure correct type at compile time
    No need for cast or Type checking

* **Important**: Cannot create generic array!

```
E[] myArray = new E[INITIAL_SIZE];                    // Error

@SuppressWarnings("unchecked")
E[] myArray = (E[]) new Object[INITIAL_SIZE];   // Ok
```

Project 1

Project 1

- **Phase A**
  - Implement Stack ADT: Stores double
    - Implement DStack
    - Using Array (ArrayStack)
    - Using Linked List (ListStack)
- **Phase B**
  - Implement Stack ADT: Use generic
    - Implement GStack
    - Using Array (GArrayStack)
    - Using Linked List (GListStack)

Project 1

- **Reverse.java**

  - Handles all music stuff
  - No need to edit for part A
  - Reverses in.dat file and writes it to out.dat
  - Accepts 4 command line arguments

    Stack Implementation: array or list
    Content type: double or generic
    Input file name: ex) in.dat
    Output file name: ex) out.dat
  - Eclipse: Run with Command line arguments

Project 1

- **Sound Exchange (SOX)**

  - Converts .wav file to .dat file & vice versa
    Reverse.java needs .dat file
    You need .wav file to play sound

  - Installed on lab machines

  - Use in command prompt / terminal
    ex) sox secret.wav secret.dat

Style Guide

- **Style Points are up to 1/3 of your grade!!**
  - Grade breakdown: ~1/3 correctness, ~1/3 write up, ~1/3 style

- **Make sure you read style guides**
  - Style guide: http://courses.cs.washington.edu/courses/cse332/14sp/projects/style.txt
  - Comment guide: http://courses.cs.washington.edu/courses/cse332/14sp/projects/commenting.pdf
  - Java Convention: http://www.oracle.com/technetwork/java/codeconv-138413.html

We DO take points off for style!

- **Make sure your code compiles!!**
  - No correctness points if your code doesn't compile!!
  - Use default package or be sure to **take out package statement**
- **Comment your code**
- **Follow Java convention:** this.isEmpty() //☹
- **Use descriptive variable / method names**
  - If variable points to beginning of queue, name it something like 'front' or 'start', not 'w' or 'g'
- **Use visibility specifiers (private/public etc.)**
  - On every classes, methods, fields. Do not just omit these!

- **Initialize all non-static fields in constructor**

```
private int size = 0; //☹
private int size; public Queue() { size = 0; } //☺
```

- **Make your code as concise as possible**

  Especially for push in ListStack – This can be done in one line!

- **Use @Override when overriding**

- **Do not leave any warning-generating code**
  - Suppress warnings only when you know exactly why it is there and why it is unavoidable
  - Suppress warnings on method/variable, but not on whole class

- **Use constants for fixed constants**

```
private static final int INITIAL_CAPACITY = 10;
private static final int RESIZE_FACTOR = 2;
```

- **Use Boolean zen**

```
if(size==0){ return true; }else{ return false; }//☹
return size == 0;    //☺
```

- **Maximize code reuse, minimize redundancy**

  - e.g. Re-use methods like `isEmpty()` instead of directly testing if size == 0 or whatever `isEmpty()` does – also improves readability

Note: a good compiler/run-time will **in-line** short methods so there is no loss in efficiency in doing this and it makes the code more readable.

Bugs & Testing

Bugs & Testing

- **Why Testing?**

  Bugs can be costly

  - Cost points in homework
  - Can cost $$$ and even life (Therac-25)

  Interesting Bug References

  - List of bugs http://en.wikipedia.org/wiki/List_of_software_bugs
  - History's worst http://www.wired.com/software/coolapps/news/2005/11/69355?currentPage=all
  - Bugs of the month http://www.gimpel.com/html/bugs.htm

Bugs & Testing

- **Reverse.java**
  does not test your stack!!

  - Stack can still have lots of bugs
    when working perfectly with Reverse.java

  - Some extreme case in past quarter:
    it only worked with Reverse.java
    (Not a good stack!)

Bugs & Testing

- **Tips for Testing**

  - Make sure program meets the spec

  - Check against Java's Stack

  - Test if each method works independently

  - Test if methods work together

  - Test for edge cases

Bugs & Testing

- **Make sure program meets the spec**
  - What is wrong with this implementation?

```
public class ListStack implements DStack {
        private LinkedList<Double> myStack;

        public ListStack() {
                myStack = new LinkedList<Double>();
        }
        public void push(double d) {
                myStack.add(d);
        }
        …
}
```

Bugs & Testing

- **Test for edge cases**

  - Empty stack

  - Push after resizing

  - Anything else?

Bugs & Testing

- **Testing tools:** JUnit Testing

  - Not required for Project 1

  - Required for Project 2

  - Covered in section later

Homework Tips

Homework Tips

- **Problem #1**

  - Use formula in the book
    (You don't have to derive it by yourself)

- **Problem #2**

  - Use following rules:

    1. $\left\lfloor \frac{\lfloor \frac{x}{m} \rfloor}{n} \right\rfloor = \lfloor \frac{x}{mn} \rfloor$ which means $\left\lfloor \frac{\lfloor \frac{n}{2} \rfloor}{2} \right\rfloor = \lfloor \frac{n}{2^2} \rfloor$

    2. $\lfloor x \rfloor = m$ if and only if $m \leq x < m + 1$

Homework Tips

- **Problem #3**

  - f(n) x 10^{-6} sec t sec, solve for n

- **Problem #4**

  - Use definitions and show you can/cannot find the constant c

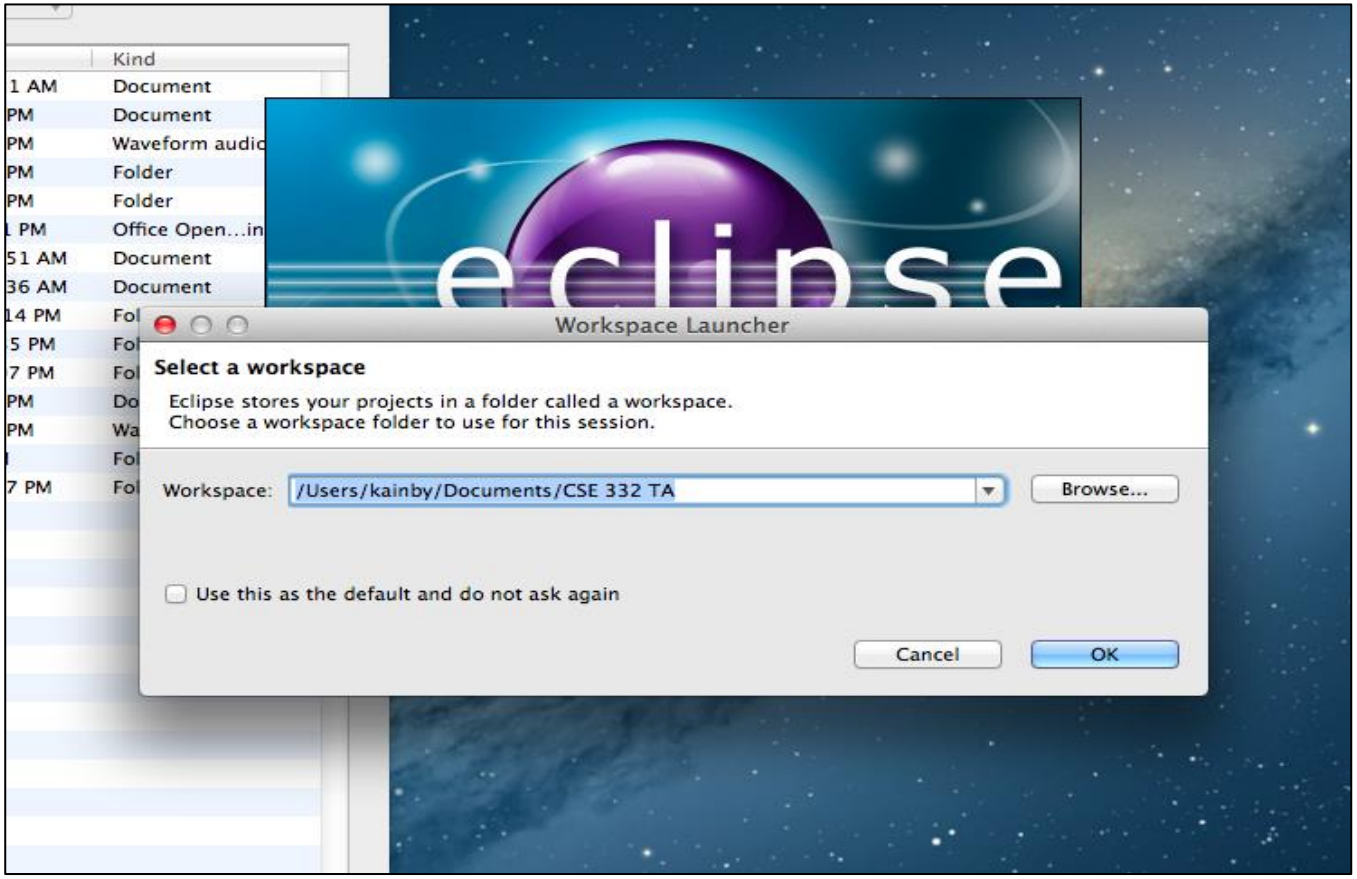
- **Problem #5**

  - Analyze runtime of each loop & merge
  - Think about maximum iteration of each loop

Eclipse

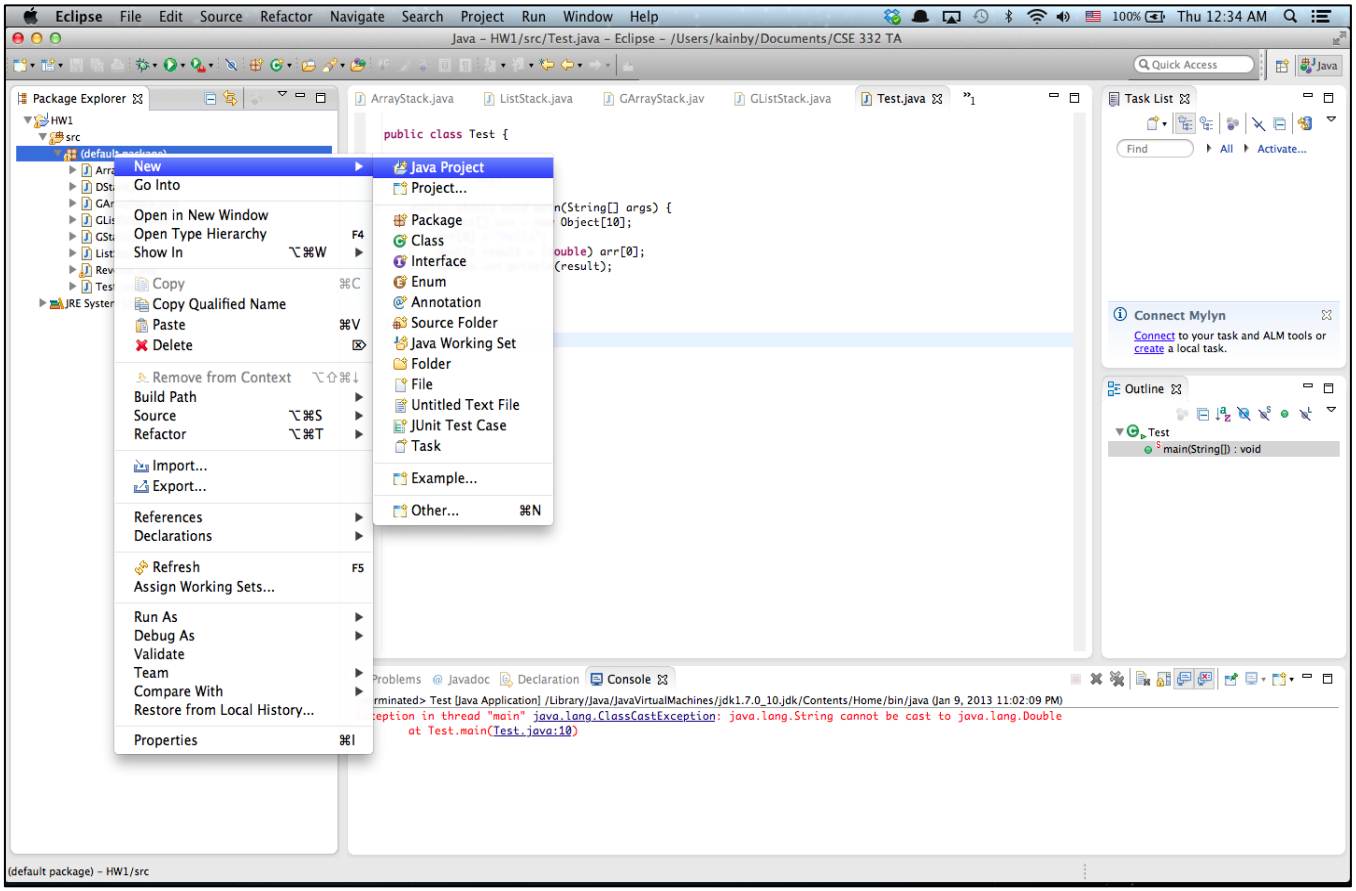
Eclipse Tutorial

- **Select WorkSpace**



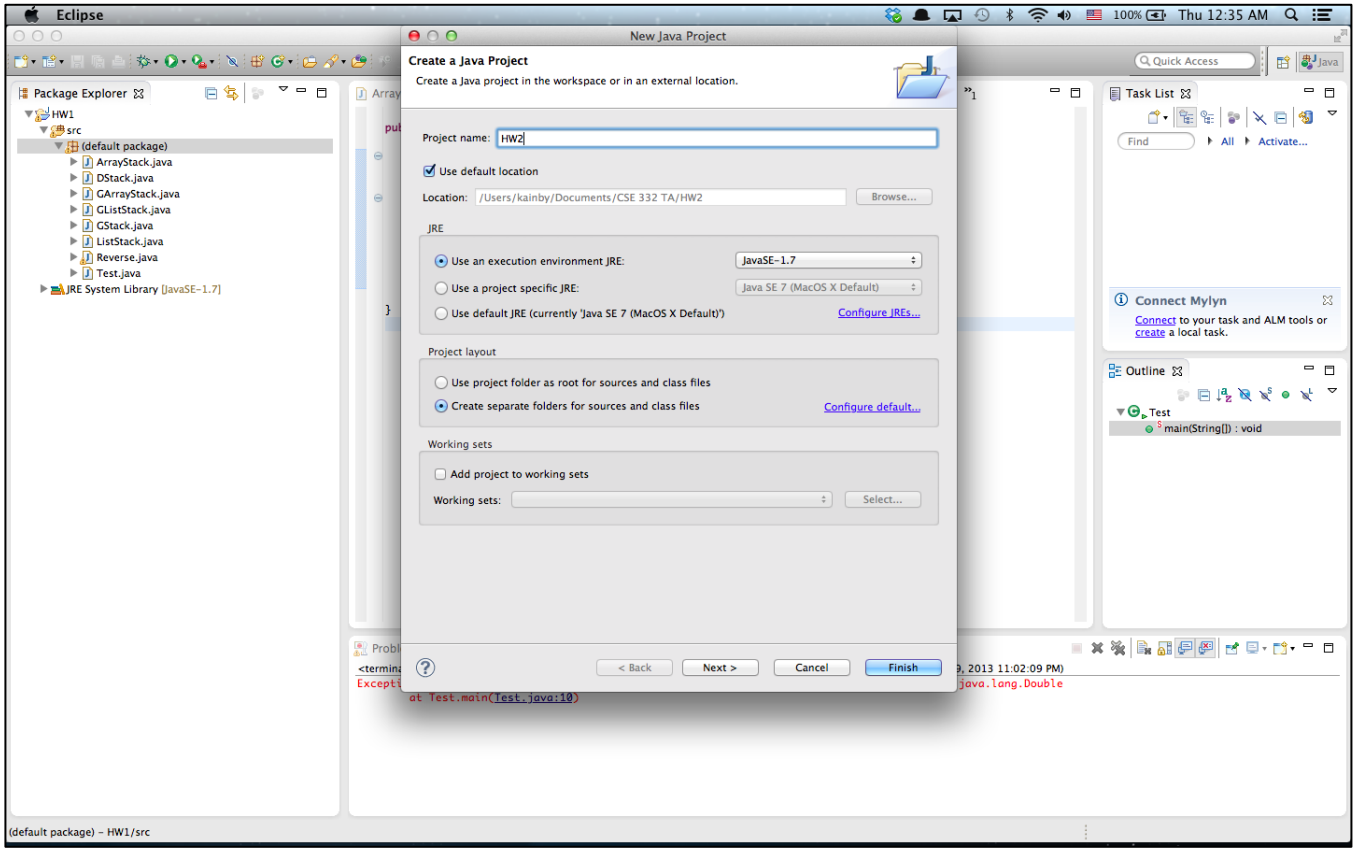
Eclipse Tutorial

- **Create Project**



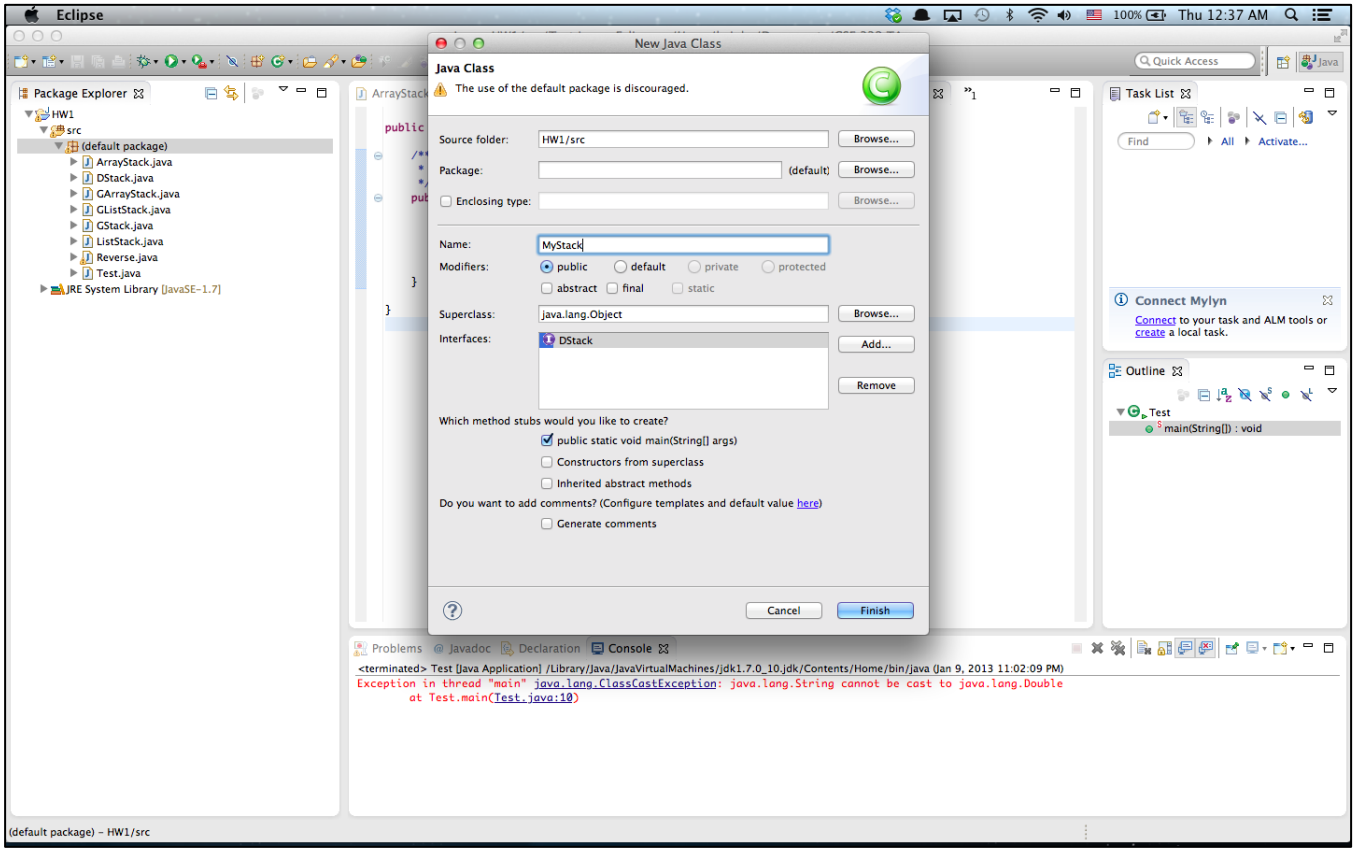
Eclipse Tutorial

- **Create Project**



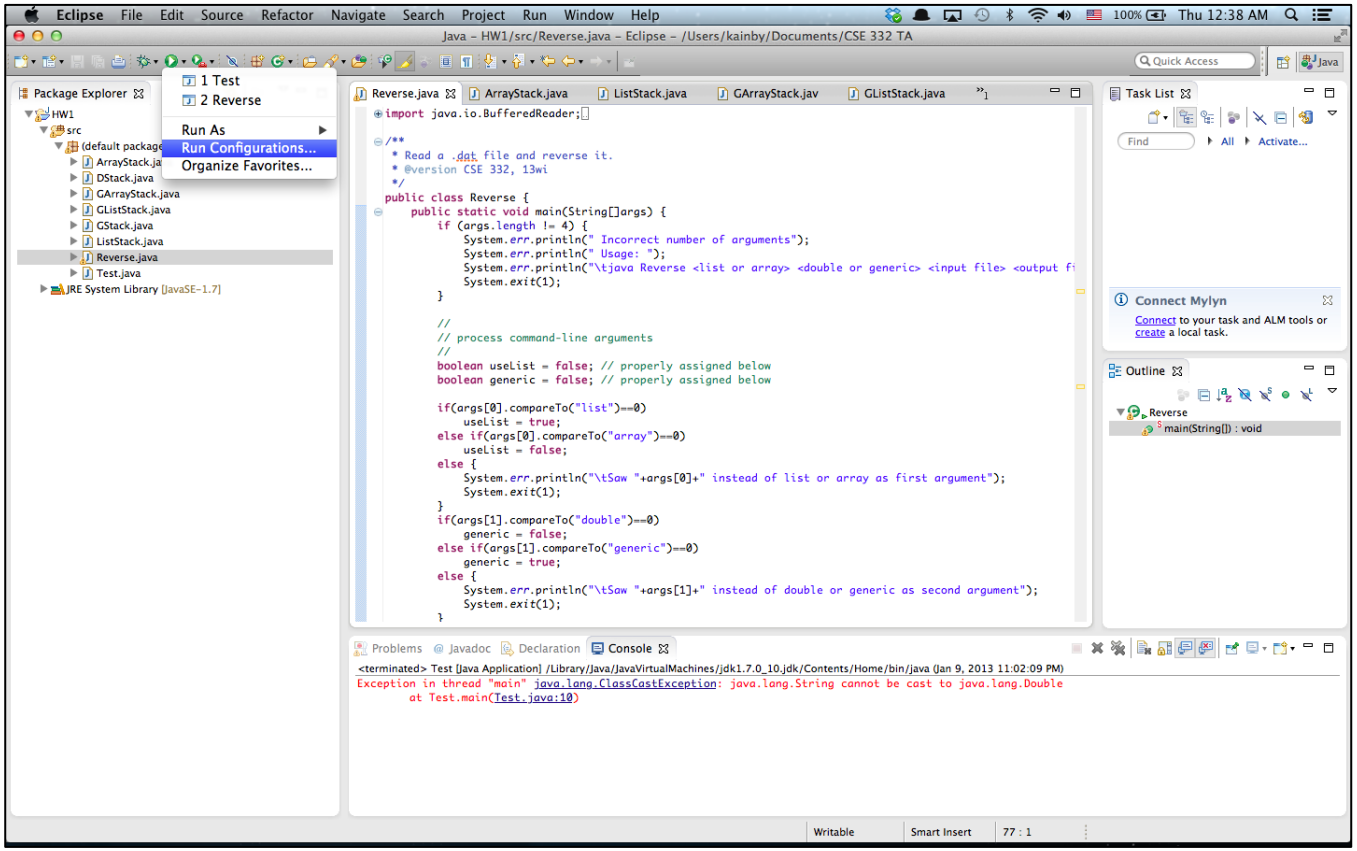
Eclipse Tutorial

- **Create Class**



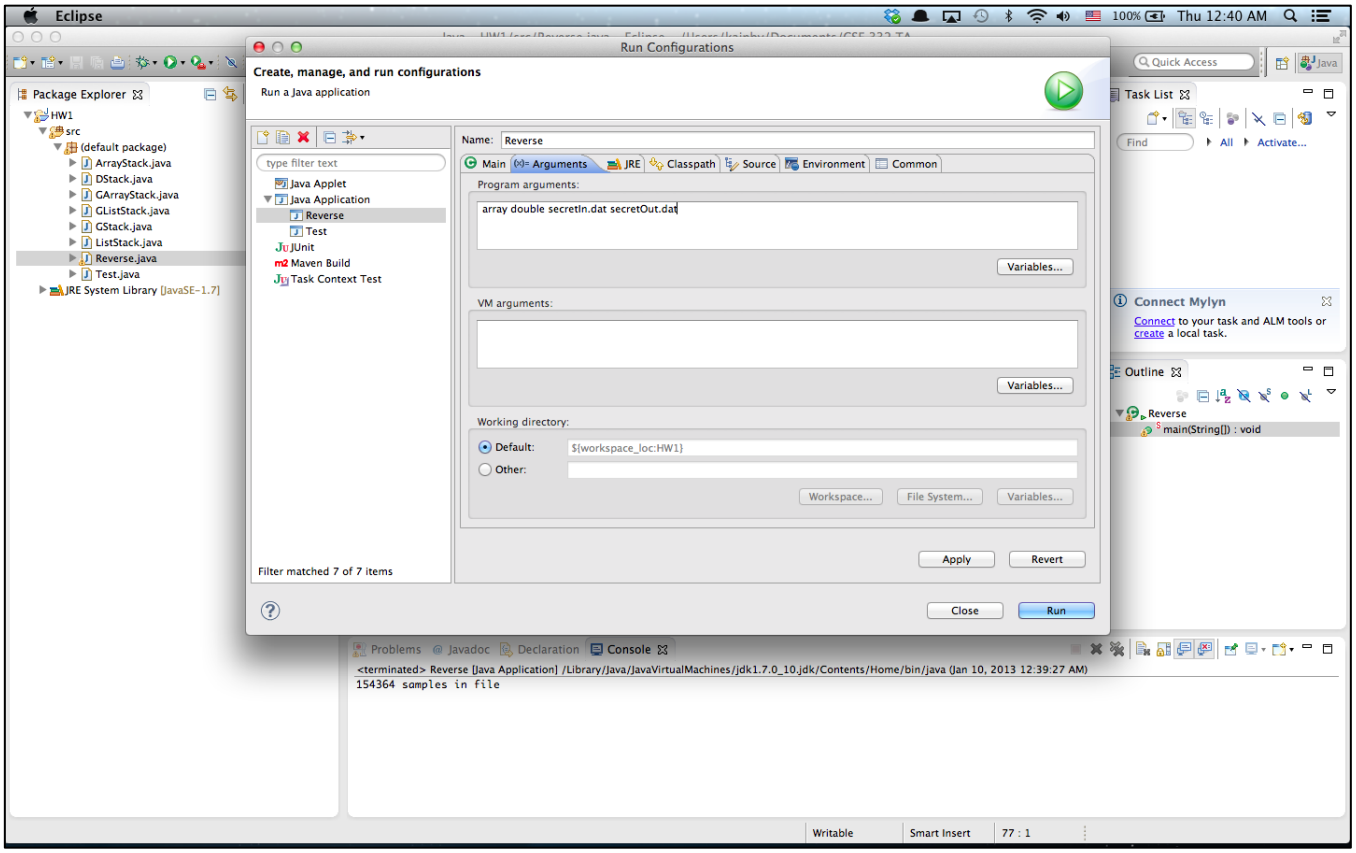
Eclipse Tutorial

- **Run Configuration (Command line Args)**



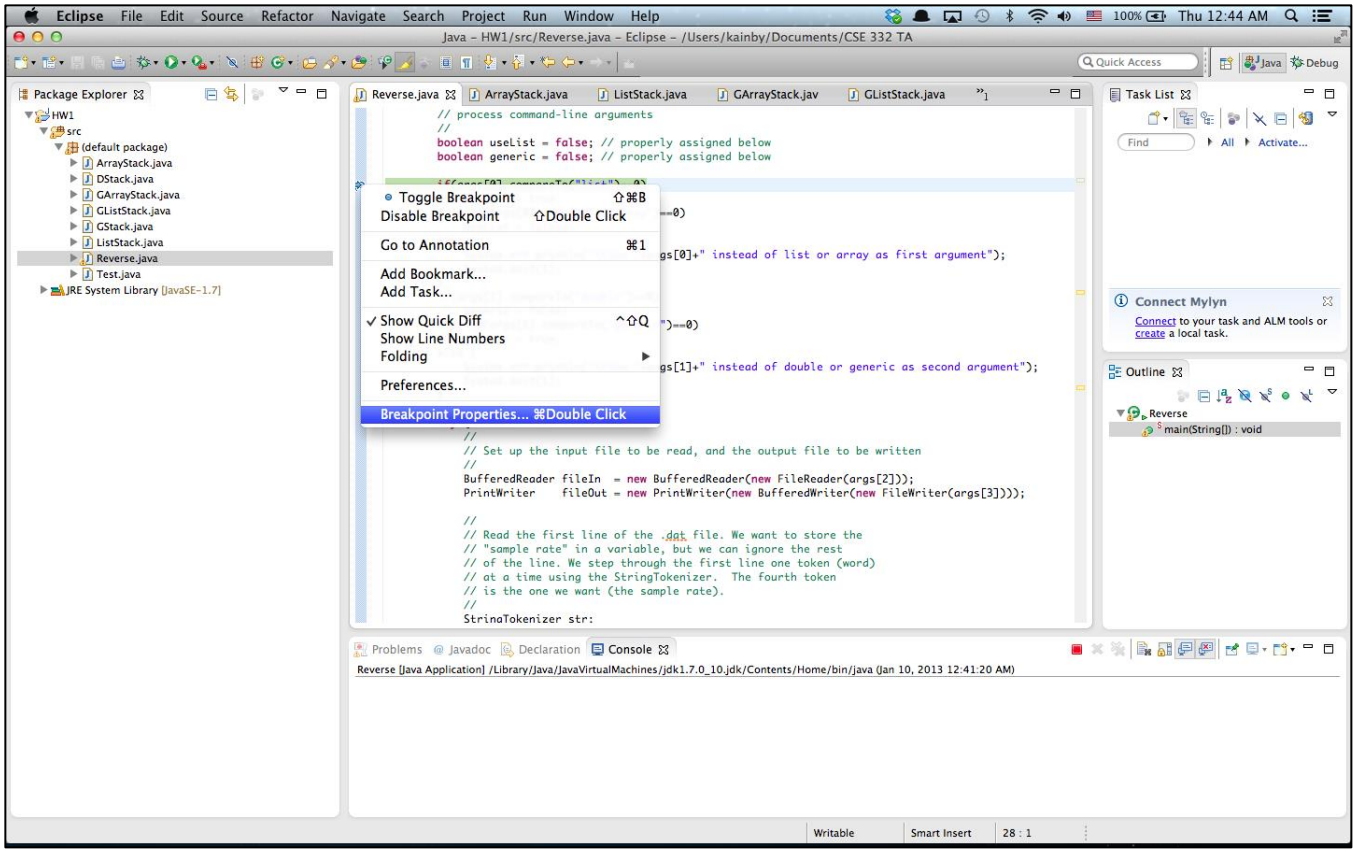
Eclipse Tutorial

- **Run Configuration (Command line Args)**



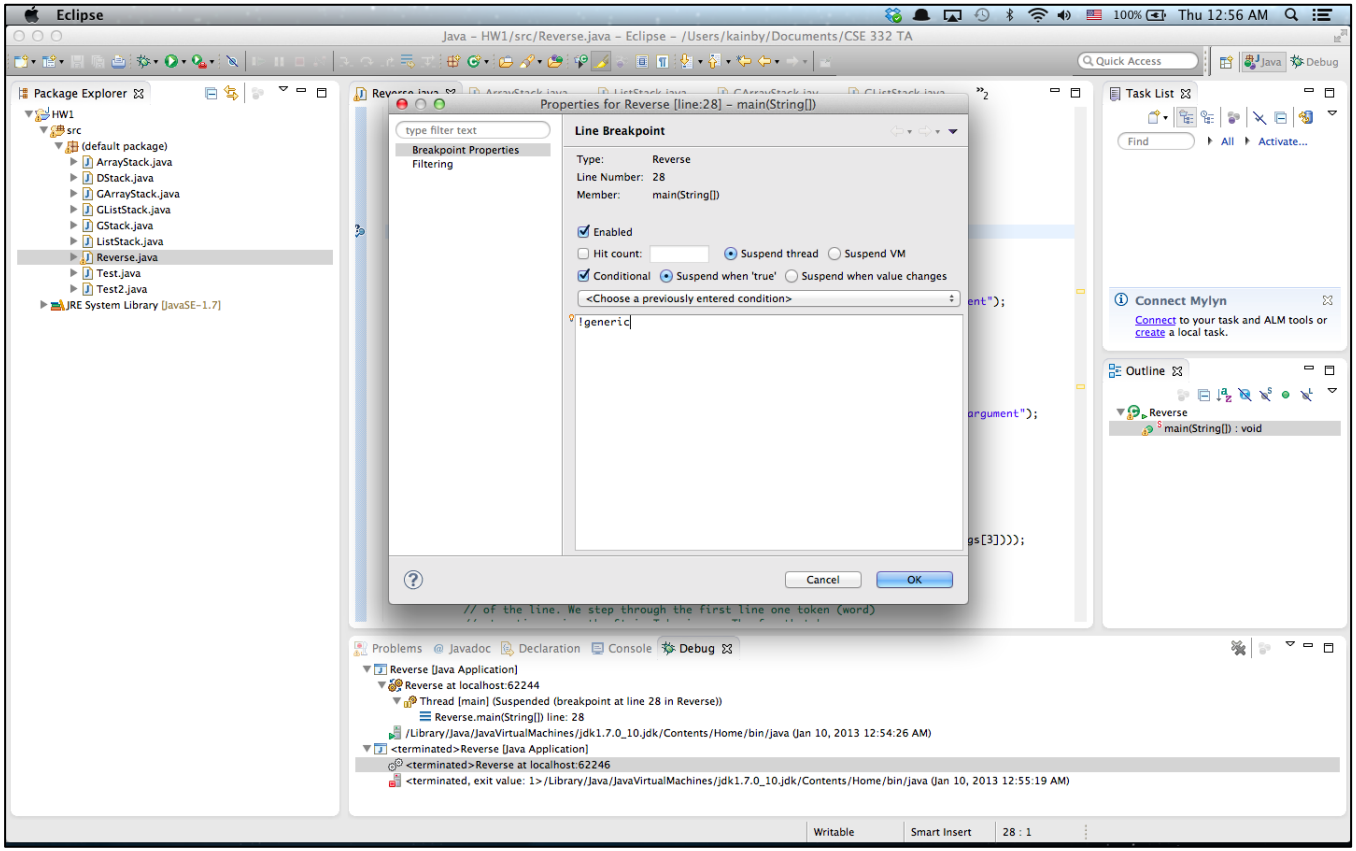
Eclipse Tutorial

- **Conditional Debugging**



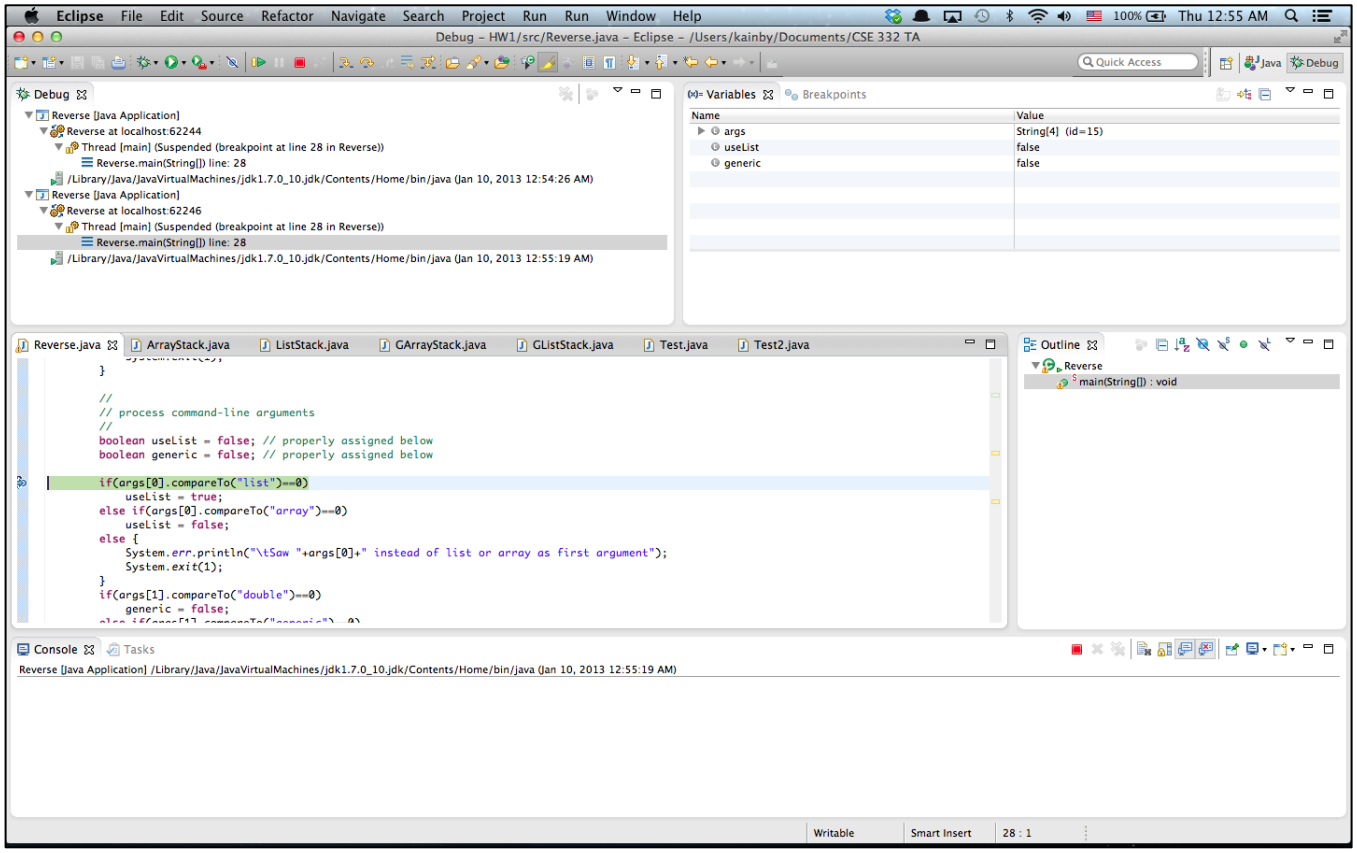
Eclipse Tutorial

- **Conditional Debugging**



Eclipse Tutorial

- **Conditional Debugging**



Eclipse Tutorial

- **More Tutorials**

  - Written Tutorial

    http://www.vogella.com/articles/Eclipse/article.html

  - Video Tutorial

    http://eclipsetutorial.sourceforge.net/totalbeginner.html

  - Eclipse Shortkeys

    http://www.rossenstoyanchev.org/write/prog/eclipse/eclipse3.html