CSE332: Data Abstractions

Section 1

Hye In Kim

Winter 2014

Section Agenda

- Recurrence Relations
- HW1 Tips
- Generics
- Project 1
- Eclipse Tutorial

Recurrence Relations

Recurrence Relations

- **Recursively defines a Sequence**

  - Example: $T(n) = T(n-1) + 3, \quad T(1) = 5$

    ^ Has $T(x)$ in definition

- **Solving Recurrence Relation**

  - Eliminate recursive part in definition

    = Find “Closed Form”

  - Example: $T(n) = 3n + 2$

Recurrence Relations

- **Expansion Method example**

  - Solve $T(n) = T(n-1) + 2n - 1, \quad T(1) = 1$

$$T(n) = T(n-1) + 2n - 1$$
$$T(n-1) = T([n-1]-1) + 2[n-1] - 1$$
$$= T(n-2) + 2(n-1) - 1$$
$$T(n-2) = T([n-2]-1) + 2[n-2] - 1$$
$$= T(n-3) + 2(n-2) - 1$$

Recurrence Relations

- **Expansion Method example**

$$T(n) = T(n-1) + 2n - 1$$
$$T(n-1) = T(n-2) + 2(n-1) - 1$$
$$T(n-2) = T(n-3) + 2(n-2) - 1$$

$$T(n) = [T(n-2) + 2(n-1) - 1] + 2n - 1$$
$$= T(n-2) + 2(n-1) + 2n - 2$$
$$T(n) = [T(n-3) + 2(n-2) - 1] + 2(n-1) + 2n - 2$$
$$= T(n-3) + 2(n-2) + 2(n-1) + 2n - 3$$

Recurrence Relations

- **Expansion Method example**

$$T(n) = T(n-1) + 2n - 1$$
$$T(n) = T(n-2) + 2(n-1) + 2n - 2$$
$$T(n) = T(n-3) + 2(n-2) + 2(n-1) + 2n - 3$$
$$\dots$$
$$T(n) = T(n-k) + [2(n-(k-1)) + \dots + 2(n-1) + 2n] - k$$
$$= T(n-k) + [2(n-k+1) + \dots + 2(n-1) + 2n] - k$$

Recurrence Relations

- **Expansion Method example**

$T(n) = T(n-k) + [2(n-k+1) + \ldots + 2(n-1) + 2n] - k$

When expanded all the way down, $T(n-k) = T(1)$

$n-k = 1$, $k = n-1$

$T(n) = T(n-[n-1]) + [2(n-[n-1]+1) + \ldots + 2(n-1) + 2n] - [n-1]$

$= T(1) + [2(2) + \ldots + 2(n-1) + 2n] - n + 1$

Recurrence Relations

- **Expansion Method example**

$$
\begin{aligned}
T(n) &= T(1) + [2(2) + \ldots + 2(n-1) + 2n] - n + 1 \\
&= T(1) + 2[2 + \ldots + (n-1) + n] - n + 1 \\
&= T(1) + 2[(n+1)(n/2) - 1] - n + 1 \\
&= T(1) + (n+1)(n) - 2 - n + 1 \\
&= T(1) + (n^2+n) - n - 1 \\
&= T(1) + n^2 - 1 \\
&= 1 + n^2 - 1 \\
&= n^2
\end{aligned}
$$

Recurrence Relations

- **Expansion Method example** **Check it!**

$T(n) = T(n-1) + 2n - 1, \quad T(1) = 1$

$T(n) = n^2$

$T(1) = 1$ same as 1^2

$T(2) = T(1) + 2(2) - 1 = 4$ same as 2^2

$T(3) = T(2) + 2(3) - 1 = 9$ same as 3^2

$T(4) = T(3) + 2(4) - 1 = 16$ same as 4^2

Recurrence Relations

- **For Homework**

  Remember to show steps!!

  - Correct answer with no steps gets no credit

  a) Show at least 2 expansions of T(n)

  b) At least 2 representations of T(n), using a)

  c) Writing T(n) in terms of k, using b)

  d) Solve for k (show steps!!)

  e) Plug in k and get the closed form

Homework Tips

Homework Tips

- **Problem #1**

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

- **Problem #3**

  - $f(n) \times 10^{-6}$ sec $\leq$ t sec, solve for n

Homework Tips

- **Problem #4**

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

- **Problem #5**

  - Analyze runtime of each loop & merge when appropriate
  - Practice finding exact runtime when you can
  - Think about maximum iteration of each loop

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 b = new Bag<String>();
b.setItem( "How about that?" );
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 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
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 parameters

    Stack Implementation: array or list
    Content type: double or generic
    Input file name: ex) in.dat
    Output file name: ex) out.dat

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://www.cs.washington.edu/education/courses/cse332/13au/projects/style.txt
  - Comment guide: http://www.cs.washington.edu/education/courses/cse332/13au/projects/commenting.pdf
  - Java Convention: http://www.oracle.com/technetwork/java/codeconvtoc-136057.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**

- **Use descriptive variable / method names**
  - If variable points to beginning of queue, name it something like 'front', 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**

- **Use @Override when overriding**

- **Do not leave warning-generating code**
  - Unless you know 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**

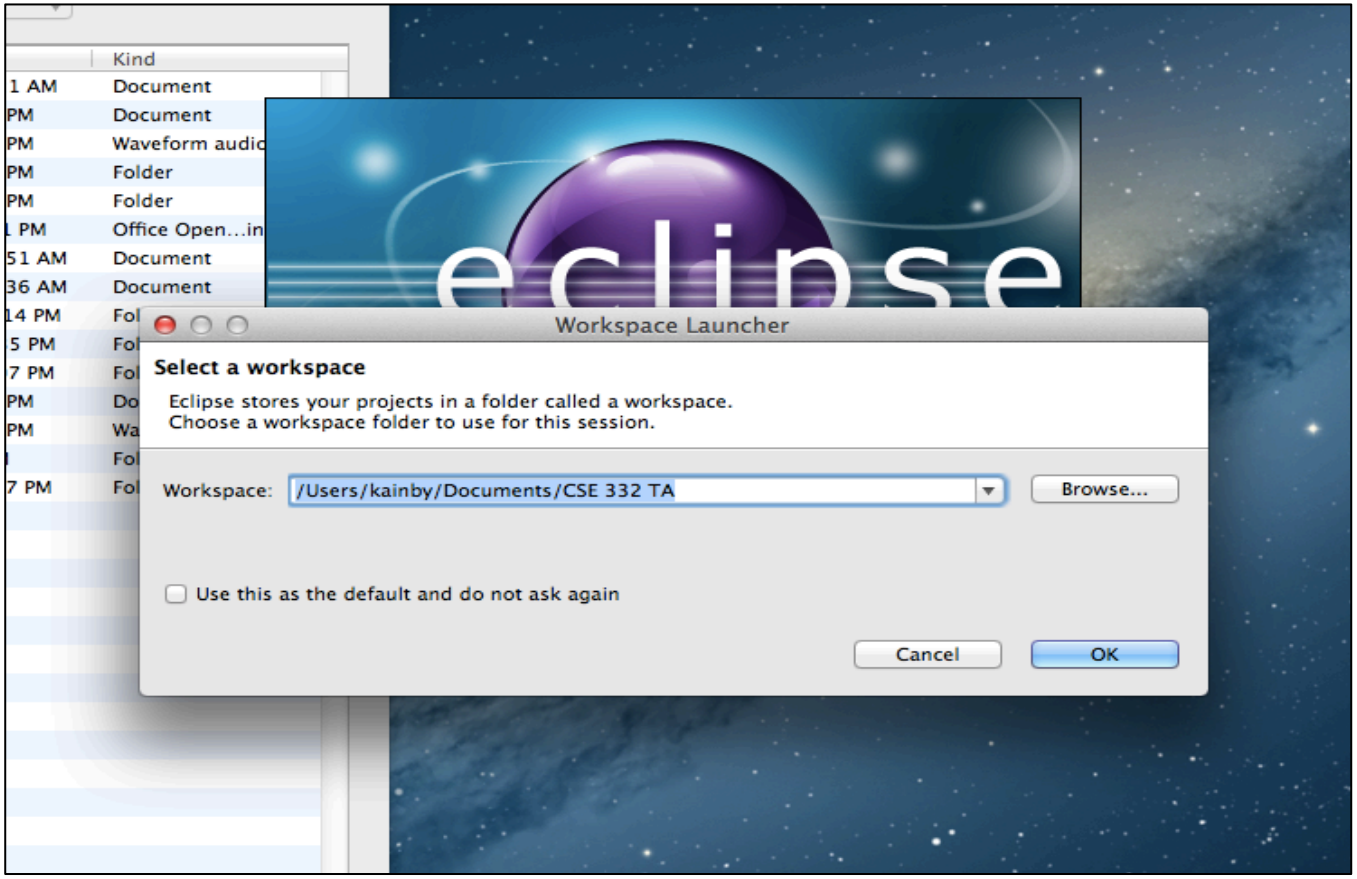
  - e.g. Re-use methods like `isEmpty()` instead of directly testing if size == 0 or whatever `isEmpty()` does – also improves readability

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.

Eclipse

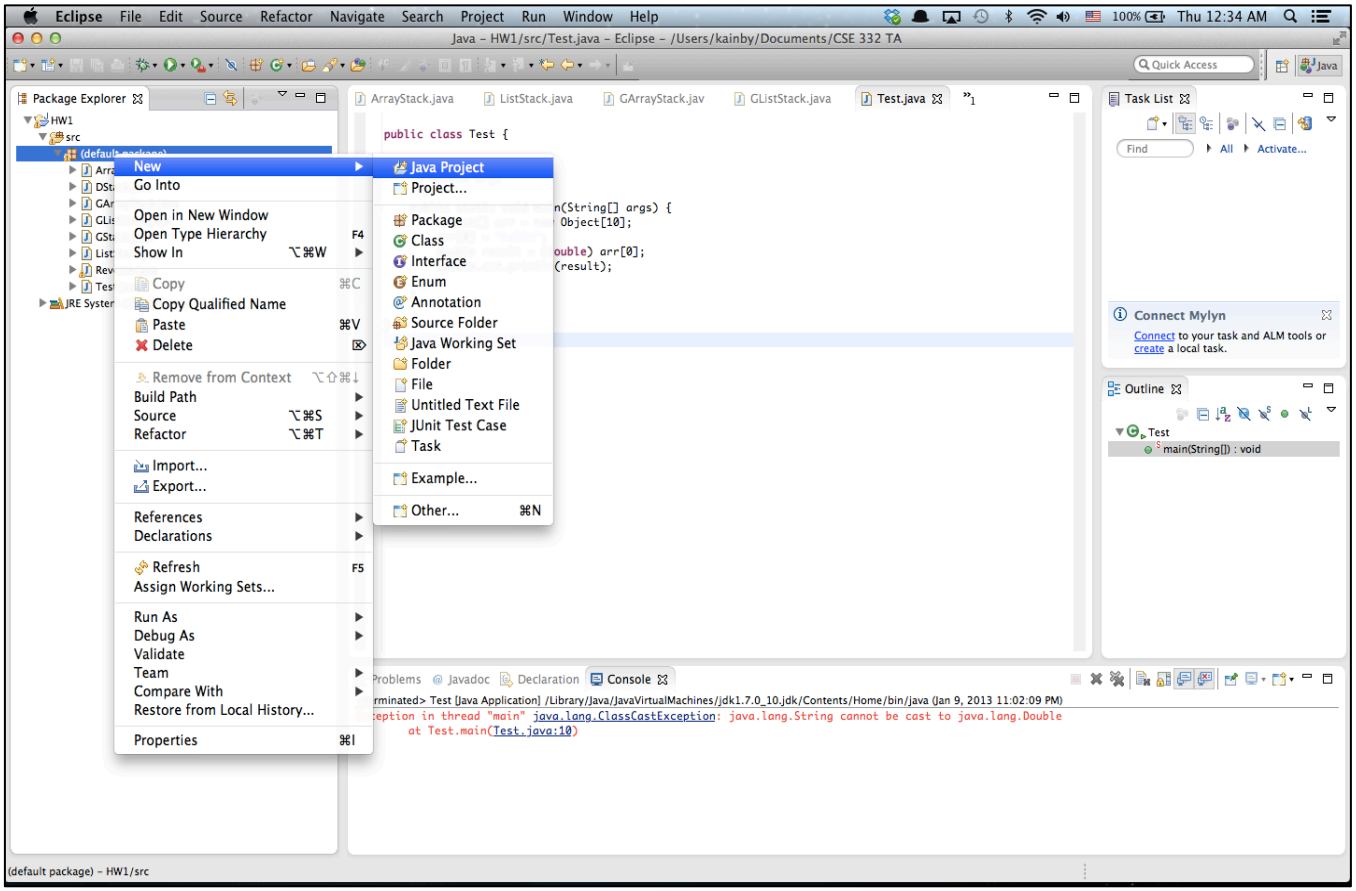
Eclipse Tutorial

- **Select WorkSpace**



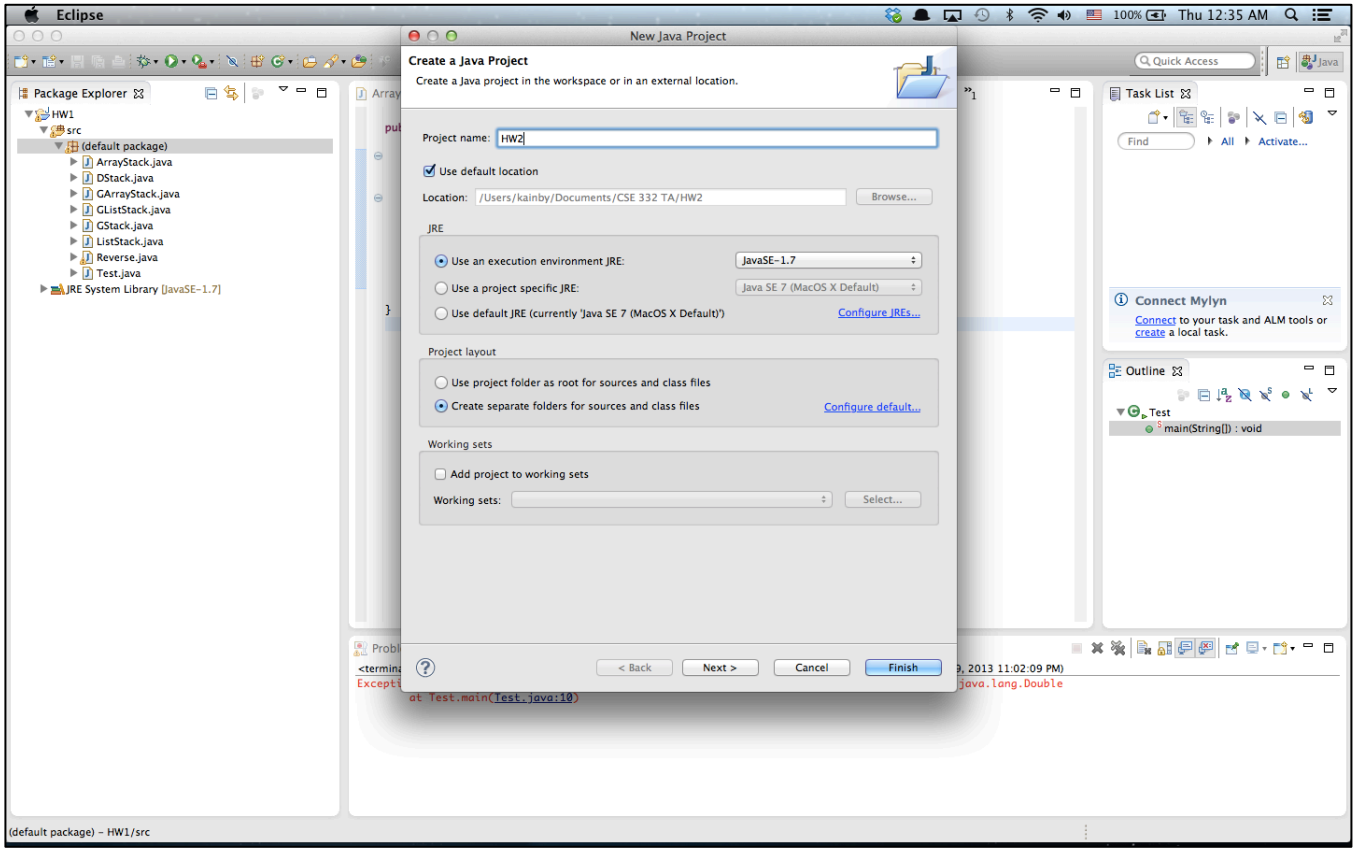
Eclipse Tutorial

- **Create Project**



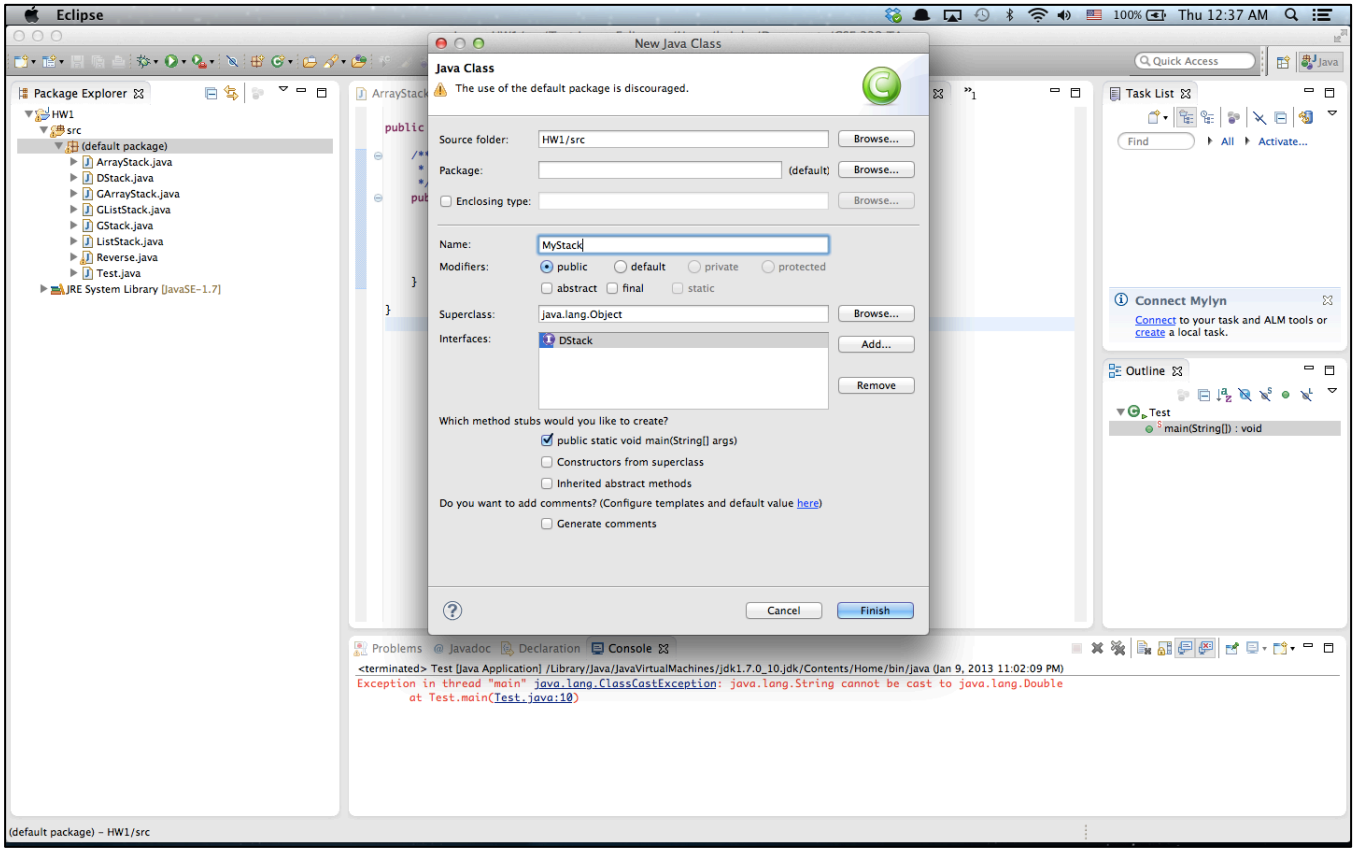
Eclipse Tutorial

- **Create Project**



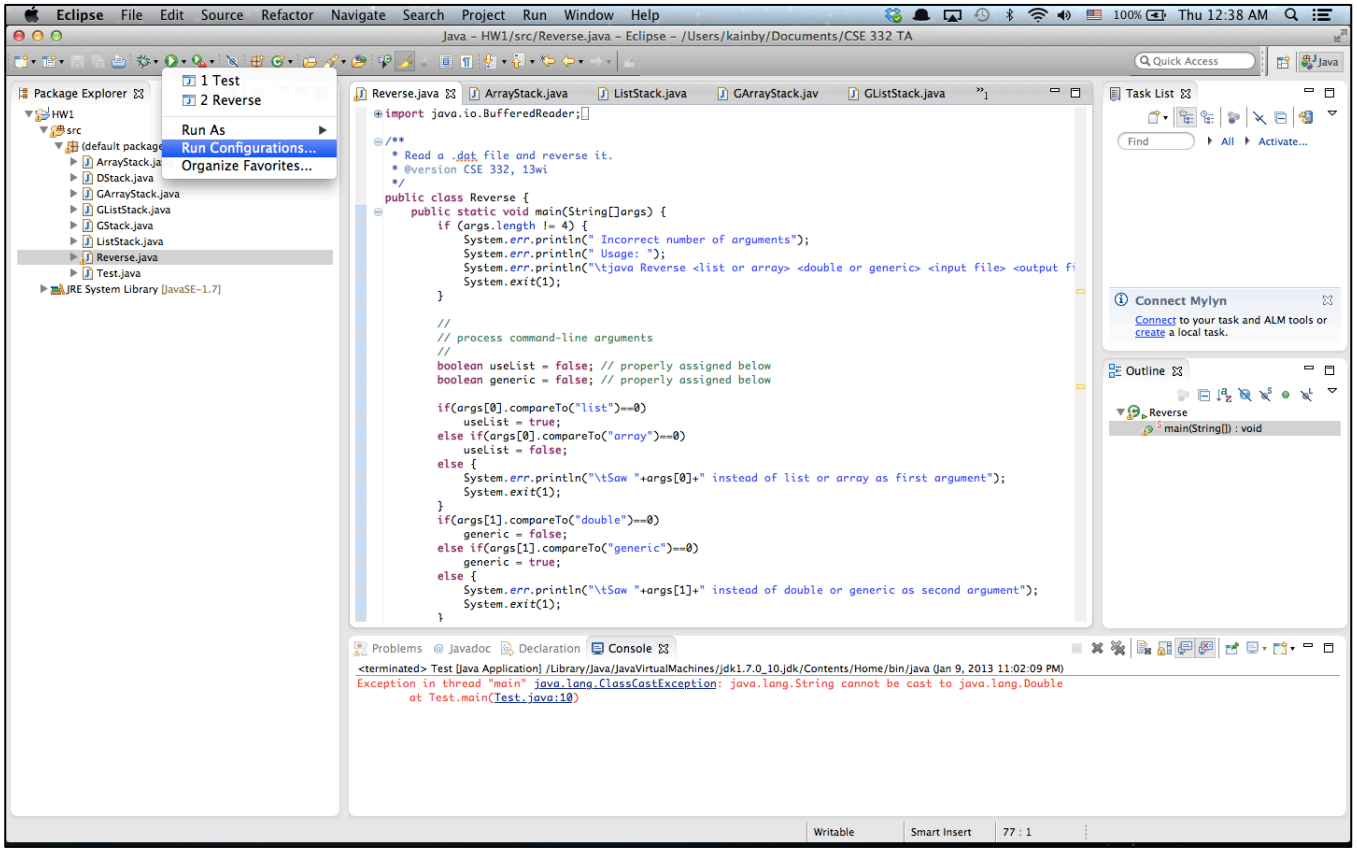
Eclipse Tutorial

- **Create Class**



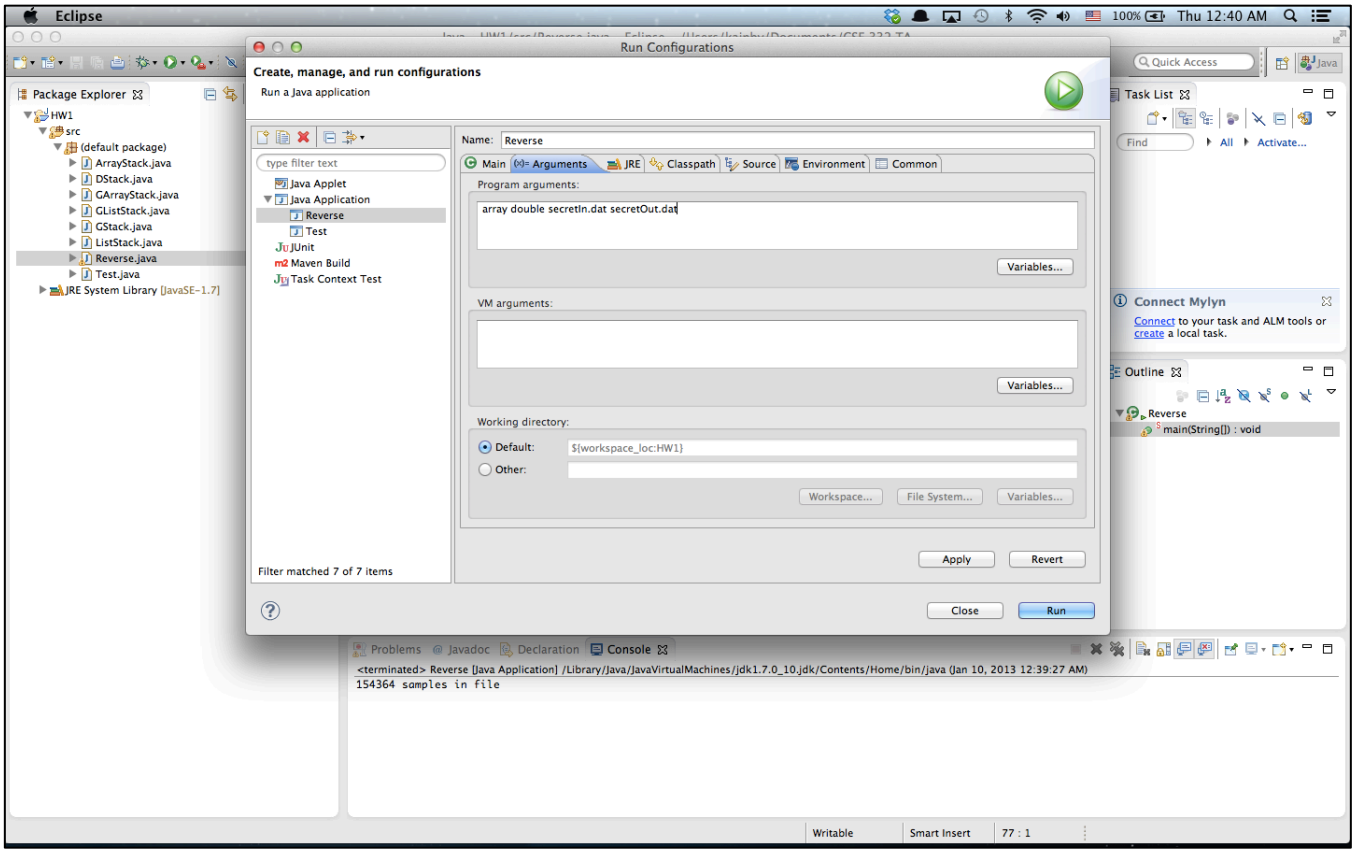
Eclipse Tutorial

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



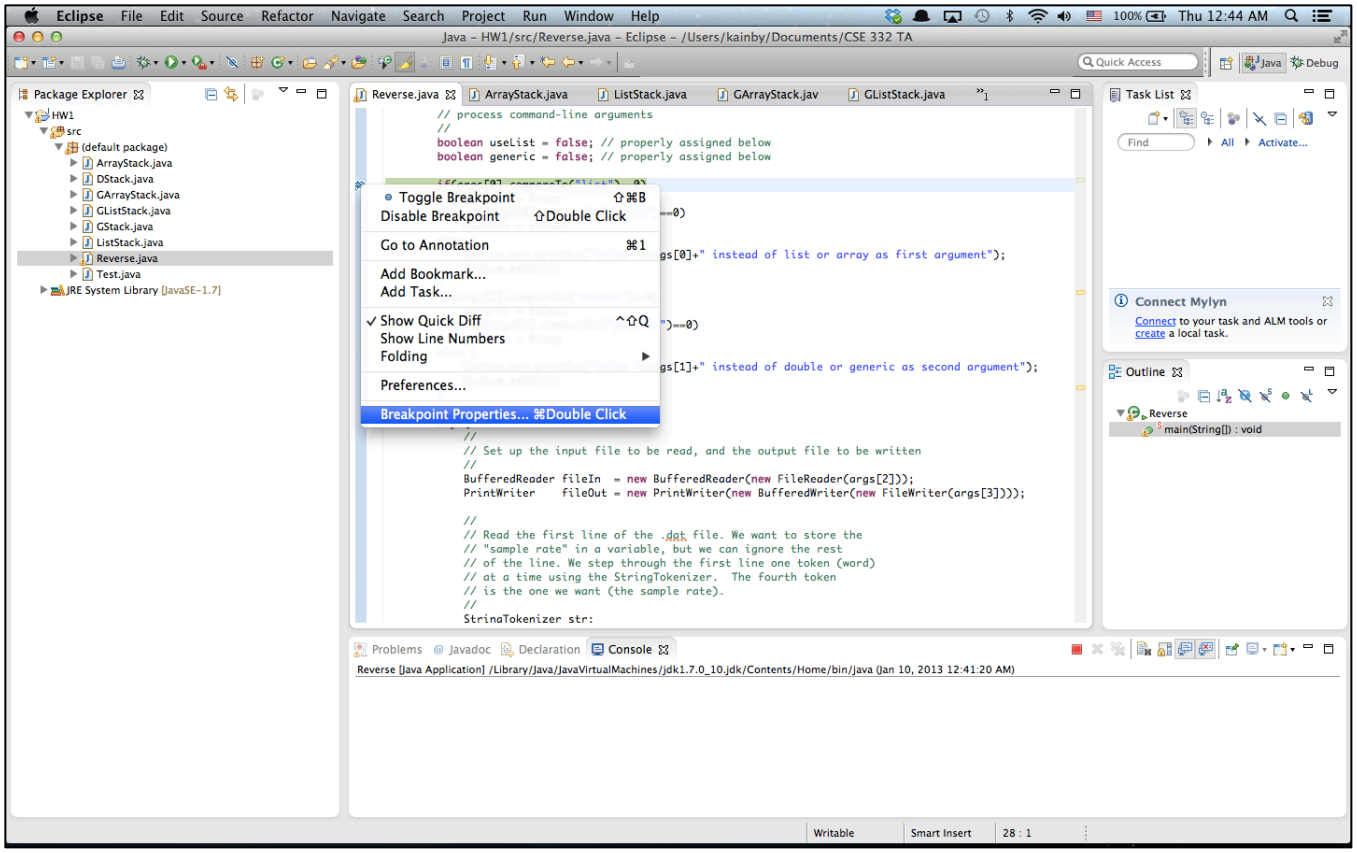
Eclipse Tutorial

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



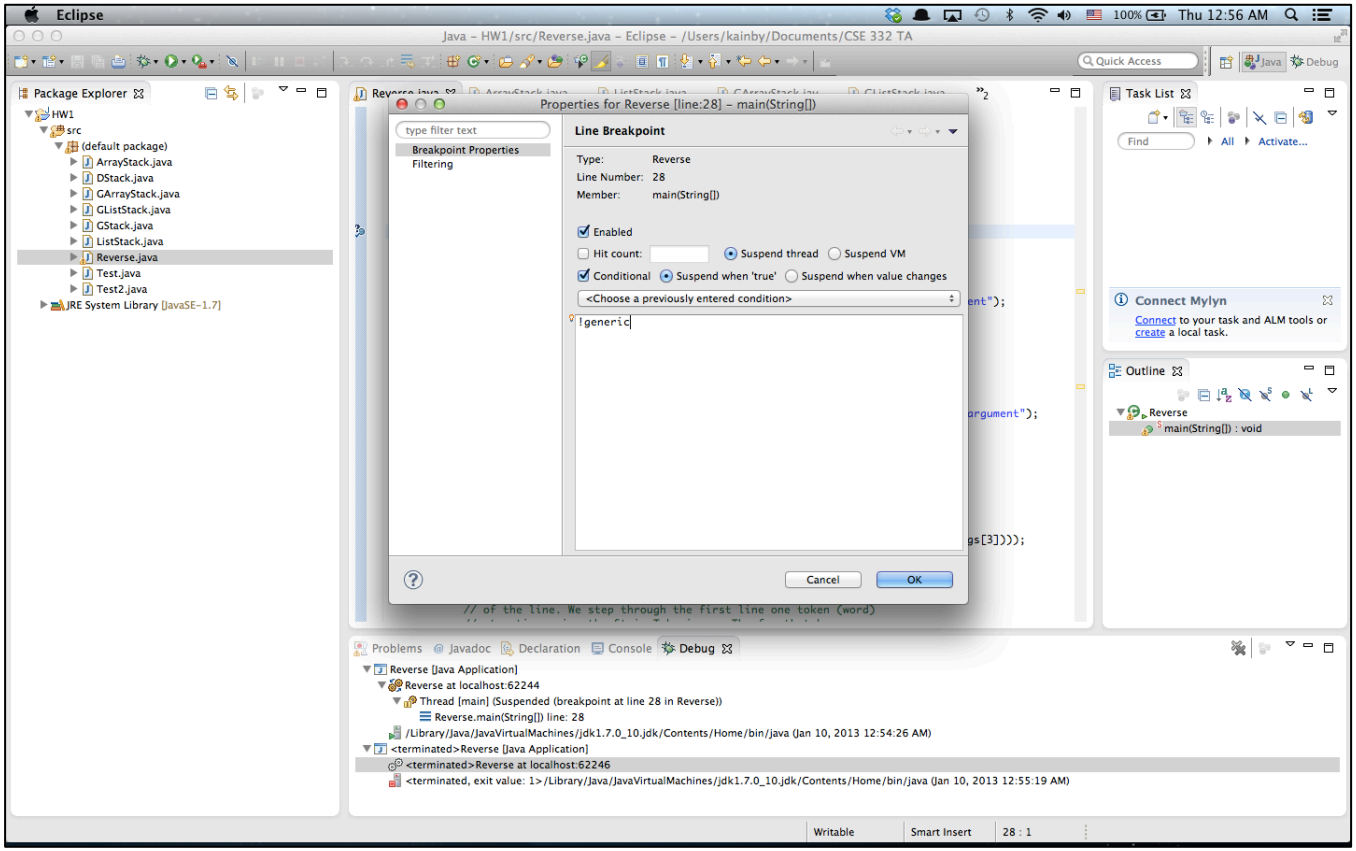
Eclipse Tutorial

- **Conditional Debugging**



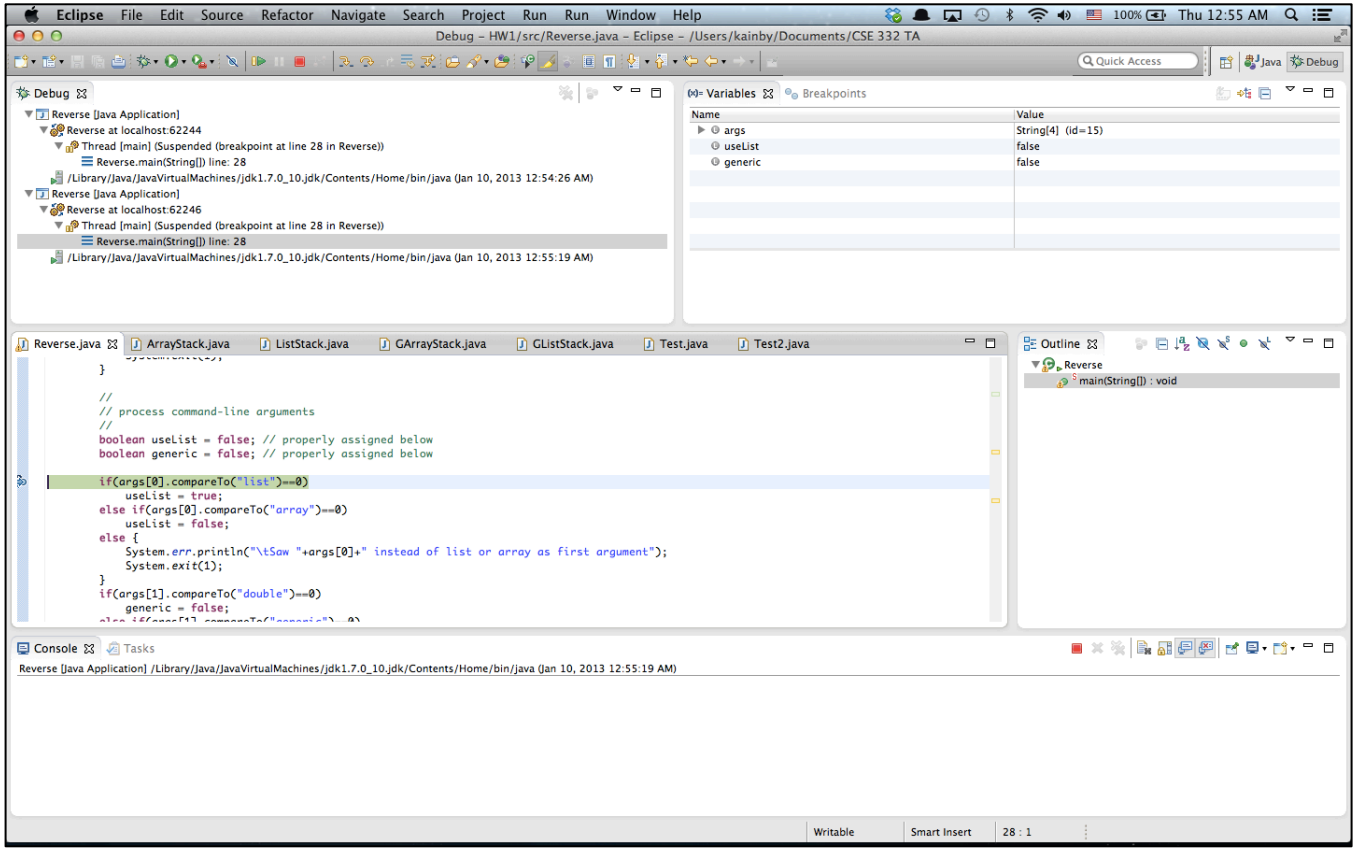
Eclipse Tutorial

- **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