Java Essentials 2013

Java Language

Main Class

```
public static void main(String[] args) {
```

Required by Class:

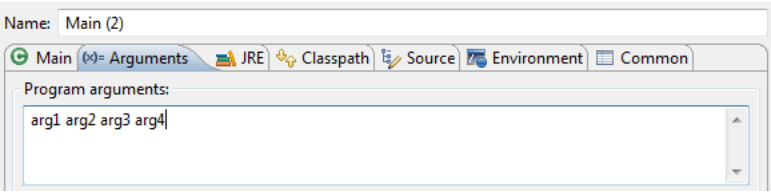
- public - can be called anywhere
- static - no instance required to run
- void - nothing returned by class
- String[] args
  - [] - an array
  - args - default variable to hold passed data
  - Passed by Java's JVM

Passing Args

Command Line

```
java Main arg1 arg2
```

In Eclipse:



Documentation

- http://docs.oracle.com/javase/6/docs/api/
- Mouseover, F2
- Windows > View > Help

Garbage Collection

- Objects referenced created in heap memory
- As long as variable referenced, it's retained
- When referenced expire, they're eligible to be garbage collected
- Garbage Collector runs own thread
- Can't force garbage collection
- OutOfMemoryError thrown if memory runs out

Expiration

- Variable to local functions or blocks expire when function is complete
- Set value to **null**

Tips for Managing Memory

- Minimize number of object created
- Runtime.*
  - Runtime.maxMemory()
  - Runtime.totalMemory()
  - Runtime.freeMemory()
- java -Xms256s HelloWorld - Initial heap size
- java -Xms256m HelloWorld - Max heap size
- java -Xms256n HelloWorld - Heap size for young generation objects

Classes and Objects

```
//Starting class has main() method
public class SimpleApplication {
        //
```

```
        public static void main(String[] args) {
                // Welcomer = datatype
                // welcomer is new instance
                Welcomer welcomer = new Welcomer();
                welcomer.sayHello();
        }
}

public class Welcomer {
      // instance variable = welcome - private to the class
      private String welcome = "Hello";
        public void sayHello() {
                System.out.println(welcome);
        }
}
```

String variables objects

String is class

```
String welcome = "Hello!";
String welcome = new String("Hello!");
      String = array[H,e,l,l,o,!]
```

Same as:

```
char[] chars = {'H','e','l','l','o','!'};
String s = new String(chars);
```

Types of Variables

- Primitives - stored directly in memory
  - Numerics - ints, floating point decimals
  - Single characters
  - Boolean (true/false)
- Complex objects
  - Strings
  - Dates
  - Everything else

Declaring a Primitive Varialbe

- Data Type - Required
- Variable name - Required
- Initial value - option
  - `int newVariable = 10;`
  - [Data type] [Variable name] = [Initial value];
  - variable name MUST start with **lowercase**
  - **Numeric default = 0**
  - **Boolean default = false**

Declaring a Complex Variable

- instance of classes
- declared in 3 parts
- Init uses **new** keyword and **class** constructor
- Inital value - optional, built from class constructor
- Date newDate = new Date();
- [Data type] [Variable name] = [Initial value from constructor]
- Variable **name alpha** char or **_**
- Date newDate = null

Scope

- local vars - declared inside function (de-referenced)
  - void doSomething()
- public vars -
  - public doSomethingElse()

Class variables = Field variable

- Declared outside a class method

```
public class MyClass {
        String sayHello = new String("Hello");
        void doSomething() {
                System.out.println(sayHello);
        }
}
```

Numeric are Primitives

- Simple value

- Stored in fastest memory
- Numeric/Boolean
- all lowercase
- int varprimitive

Type	Bits	Min	Max
byte	8	-128	127
short	16	-32,768	32,767
int	32	-2,147,483,648	2,147,483,647
long	64	-9.22337E+18	9.22337E+18
float	32		
double	64		

Setting Literal Values

- `byte b = 1;`
- `short s = 10;`
- `int i = 10;`
- `long l = 100L;`
- `float f = 150.5f;`
- `double d = 150.5d;`

Without extra char, Java will cast it automatically, not using memory most efficiently

Helper Classes

Data Type	Helper Class
byte	Byte
short	Short
int	**Integer**
long	Long
float	Float
double	Double

Numeric Wrapper Class

- Double provides tools for convering float values
- `double doubleValue = 156.5d;`
- `Double doubleObj = new Double(doubleValue);`
- `byte myByteValue = boubleObj.byteValue();`
- `int myIntValue = doubleObj.intValue();`
- `float myFloatValue = doubleObj.floatValue();`
- `Strig myString = double.Obj.toString();`

BigDecimal

For currency values that have guaranteed precision

Problem

```
BigDecimal payment = new BigDecimal(1115.37);
System.out.println(payment.toString());
```

>1115.36999999908796...

```
...Result determined by OS, Processor, etc
```

Solution

```
double d = 1115.737;
String ds = Double.toString(d);
BigDecimal bd = new BigDecimal(ds);
System.out.println("the value is " + db.toString());
```

>Value is 1115.37

Converting Primitives Upward/Downward

Converting upwards:

- int intvalue = 120;
- double doubleResult = intValue; // = 120.0 (double)

Converting downwards:

```
double doublevalue = 3.99;
int intResult = doubleValue; // won't even compile due to accuracy loss
int intResult = (int)doubleValue; // explicit accuracy loss by truncating
= 3
```

- double
- float
- long
- int
- short
- byte

Wrapping Around when converting Downward

- double > byte

```
int i = 128;
byte b = (byte)i;
> -128
```

Casting Syntax VS Helper Class

Casting

- Doesn't add another object to memory
- Can't use afterwards

```
double doubleValue = 3.99;
int intResult = (int)doubleValue;
```

Helper Class

- Creates instance of Helper Class in memory
- Good when needing to do things with it later

```
double doubleValue = 3.99;
double doubleObj = new Double(doubleValue);
int intResult = doubleObj.intValue();
```

Operators

- follows C standards

Types

- Assignment
- Equality or relational
- Mathematical
- Conditiona
- Ternary (short-hand conditional)

Assignment and Math Operators

=, +, -, *, /, %

Incrementing

```
intValue = 10;
intValue ++; // 11
intValue --; // 9
intValue += 5; // 15
intValue -= 5; // 5
intValue *= 5; // 25
intValue /= 5; // 2
```

PostFix vs Prefix Incrementing

int intValue = 10;

PostFix

Do operator, then evaluate
System.out.println(intValue ++)
output = 10

new value = 11

Prefix (a.k.a.) Unerary

Evaluate, then do operator
System.out.println(+ intValue);
output = 11
new value = 11

Comparing Values

>, <, >=, <=, instanceof (Class Membership)

```
String s = "Hello";
if (s instanceof java.lang.String) {
        System.out.println("s is a String");
}
```

Comparing Strings

```
String s1 = "Hello";
String s2 = "Hello";
if (s1 == s2) {
        System.out.println("They Match!);
} else {
        System.out.println("No Match!);
}
```
> No Match!

```
if (s1.equals(s2)) {
        System.out.println("They Match!);
} else {
        System.out.println("No Match!);
}
```
>They Match!

Equality Operators

`==` Equality
`!=` IInequality
`if(!this)` Reversing logic: **booleans only**

Conditional Operators

`&&` AND
`||` OR

Characters

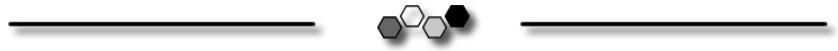
Code:

```
        public static void main(String[] args) {
           char c1 = '1';
           char c2 = '2';
           char c3 = '3';

           //unicode
           char dollar = '\u0024';

           System.out.print(dollar);
           System.out.print(c1);
           System.out.print(c2);
           System.out.println(c3);

           //Wrapper Class: Character
           char a1 = 'a';
           char a2 = 'b';
           char a3 = 'c';
           System.out.print(Character.toUpperCase(a1));
           System.out.print(Character.toUpperCase(a2));
           System.out.println(Character.toUpperCase(a3));

    }
```

Booleans

Code:

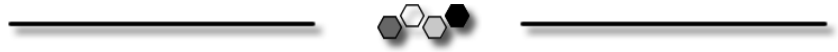
```
        public static void main(String[] args) {
           boolean b1 = true;
           boolean b2 = false;

           System.out.println("The value of b1 is " + b1);
           //true

           System.out.println("The value of b2 is " + b2);
           //false

           boolean b3 = !b1;
           System.out.println("The value of b3 is " + b3);
           // false

           int i = 0; //NOT true in java
           boolean b4 = (i != 0); //translate int to boolean
           System.out.println("The value of b4 is " + b4);
           //false

           String s1 = "true"; // or TRUE
           boolean b5 = Boolean.parseBoolean(s1);
           System.out.println("The value of b5 is " + b5);
           // true

           String s2 = "FALSE"; // or false
           boolean b6 = Boolean.parseBoolean(s2);
           System.out.println("The value of b6 is " + b6);
           // false

           String s3 = "BLAH"; // =  false
           boolean b7 = Boolean.parseBoolean(s3);
           System.out.println("The value of b7 is " + b7);
           //false
   }
```



StringOutput

Code:

```
public class Main {

   public static void main(String[] args) {
           char c = 'z';
           boolean bool = true;
           byte b = 127;
           short s = 32000;
           int i = 2000000;
           long l = 10000000L;
           float f = 1234245.435234f;
           double d = 112312312331.34;

           System.out.println(c);
           System.out.println(bool);
           System.out.println(b);
           System.out.println(s);
           System.out.println(i);
           System.out.println(l);
           System.out.println(f);
           System.out.println(d);
/*
32000
2000000
10000000
1234245.4
1.1231231233134E11
*/
           // first value in the print Math or String
           System.out.println("The value of s is " + s);
           //The value of s is 32000

           // if one String is involved, whole thing is string
           System.out.println(s + " The value of s is ");
           //32000 The value of s is

           // math converted first, then tacked onto string
           // String first, no math...all string

           // needs 'Date' package
           Date myDate = new Date();
           System.out.println("The new date is " + myDate);
           //The new date is Fri Aug 03 14:55:55 CDT 2012
   }

}
```

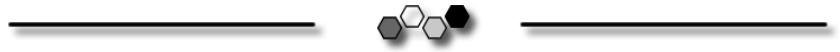


Calculator

Code:

```
import java.io.*;

public class Calculator {
```

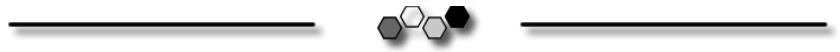
```
   public static void main(String[] args) {
           String s1 = getInput("Enter a numeric value: ");
           String s2 = getInput("Enter a numeric value: ");

           double d1 = Double.parseDouble(s1);
           double d2 = Double.parseDouble(s2);
           double result = d1 + d2;
           System.out.println("The answer is " + result);
//Enter a numeric value: 10
//Enter a numeric value: 25.5
//The answer is 35.5

/*
Enter a numeric value: xyz
Enter a numeric value: abc
Exception in thread "main" java.lang.NumberFormatException: For input string: "xyz"
   at sun.misc.FloatingDecimal.readJavaFormatString(Unknown Source)
   at java.lang.Double.parseDouble(Unknown Source)
   at Calculator.main(Calculator.java:9)
*/

   }

   private static String getInput(String prompt) {
           BufferedReader stdin = new BufferedReader(
                           new InputStreamReader(System.in));

           System.out.print(prompt);
           System.out.flush();

           try {
                   return stdin.readLine();
           } catch (Exception e) {
                   return "Error: " + e.getMessage();
           }
   }

}
```



CompareStrings

Code:

```
public class Main {

   public static void main(String[] args) {
           int monthNumber = 7;

           if (monthNumber >= 1 && monthNumber <=3) {
                   System.out.println("You're in Quarter 1");
           }
           else if (monthNumber >= 4 && monthNumber <=6) {
                   System.out.println("You're in Quarter 2");
           }
           else {
                   System.out.println("You're not in the first half of the year!");
           }
   }
}

public class CompareStrings {

   public static void main(String[] args) {
           String month = "February";

           if (month.equals("February")) {
                   System.out.println("It's the second month!");
           }

   }

}
```



Switch Statements: Integers

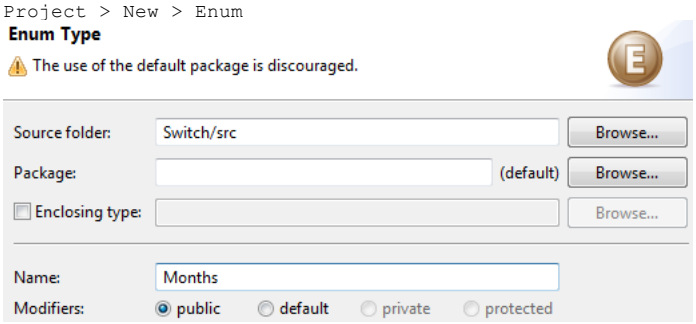
Code:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class SwitchWithInts {

   public static void main(String[] args) {
           String input = getInput("Enter a number between 1 and 12: ");
           int month = Integer.parseInt(input);

           switch (month) {
           case 1:
                   System.out.println("The month is January");
                   break; //must use break to leave switch
           case 2:
                   System.out.println("The month is February");
```

```
                break;
            case 3:
                System.out.println("The month is March");
                break;
            default:
                break;
            }
    }

    private static String getInput(String prompt) {
            BufferedReader stdin = new BufferedReader(
                            new InputStreamReader(System.in));

            System.out.print(prompt);
            System.out.flush();

            try {
                return stdin.readLine();
            } catch (Exception e) {
                return "Error: " + e.getMessage();
            }
    }

}
```

Switch Statements: Enums

Project > New > Enum



Code:

```
public enum Month {
    JANUARY, FEBRUARY, MARCH; //Constants of enum class
}


public class SwitchWithEnums {
    // Enumerations
    public static void main(String[] args) {

//            int month = 1;
            Month month = Month.FEBRUARY;

            switch(month) {
            case JANUARY:
                System.out.println("It's the first month");
                break;
            case FEBRUARY:
                System.out.println("It's the second month");
                break;
            case MARCH:
                System.out.println("Its the third month");
            }

    }

}
```

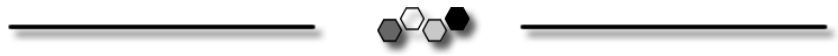
Switch Statements: Strings (Java SE7)

Loops

Code:

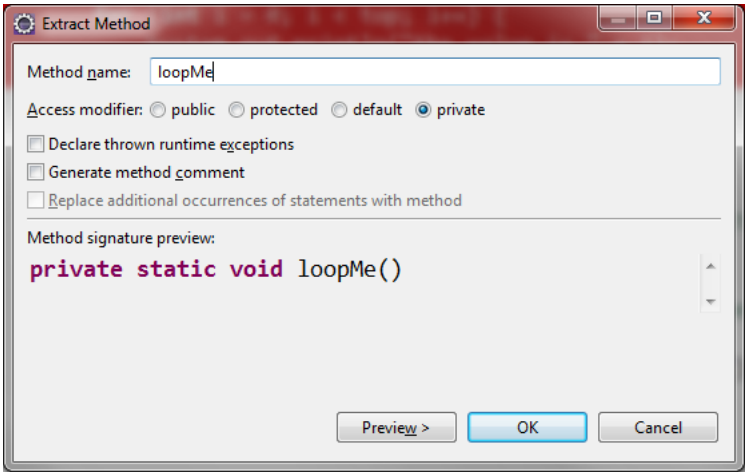
```
public class Main {

    static private String[] months =
            {"January", "February", "March",
            "April", "May", "June",
            "July", "August", "September",
            "October", "November", "December"};

    public static void main(String[] args) {
```

```
            // using counter variables instead of complex objects
//              for (int i = 0; i < months.length; i++) {
//                      System.out.println(months[i]);
//              }

            // For each month in the months[] array
//              for (String month : months) {
//                      System.out.println(month);
//              }

            // eval BEFORE loop
//              int counter = 0;
//              while (counter < months.length) {
//                      System.out.println(months[counter]);
//                      counter ++;
//              }

            // eval AFTER loop
            int counter = 0;
            do {
                    System.out.println(months[counter]);
                    counter ++;
            } while (counter < months.length);
    }
}
```



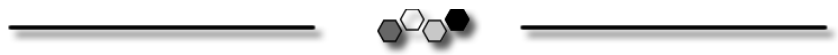
Methods

Refactoring

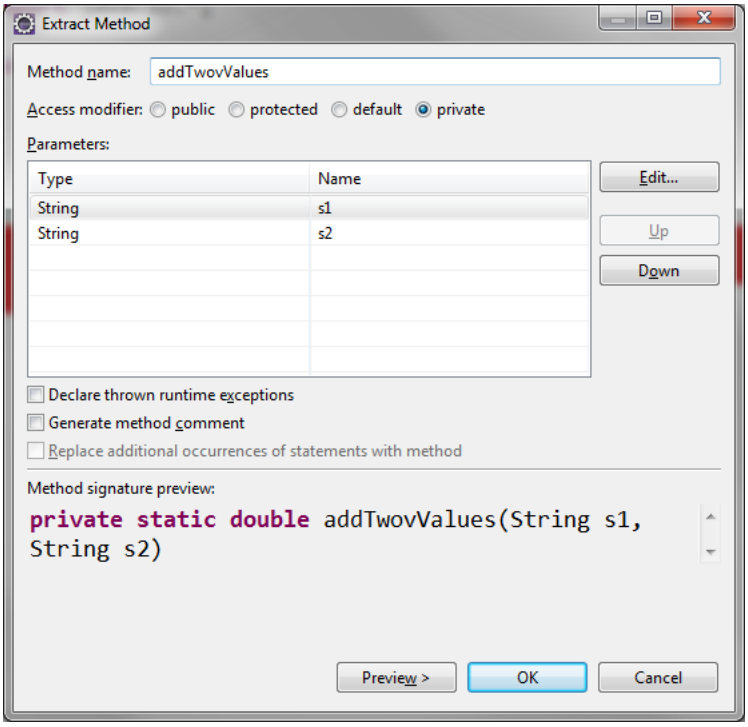


Code:

```
public class Main {

    public static void main(String[] args) {
            doSomething();
            //refactoring, copy code, Refactor..
            //will create a new method and reference it here
            loopMe();
    }

    private static void loopMe() {
            int top = 10;
            for (int i = 0; i < top; i++) {
                    System.out.println("the value is " + i);
            }
    }


    //Access modifier public, private, protected (inheritance), none (protected
package)
    //Static - class method, only used inside class
    //non-Static - used in instances
    //Static must create instance to call non-Static '.method()'

    private static void doSomething() {
            System.out.println("This method has been called");
    }


}
```



Extracting a Method



Code:

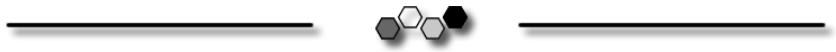
```
import java.io.*;

public class Calculator {

   public static void main(String[] args) {
           String s1 = getInput("Enter a numeric value: ");
           String s2 = getInput("Enter a numeric value: ");
// Extracting a Method
           double result = addTwovValues(s1, s2);

           System.out.println("The answer is " + result);
   }

// Extracted Method
   private static double addTwovValues(String s1, String s2) {
           double d1 = Double.parseDouble(s1);
           double d2 = Double.parseDouble(s2);
           double result = d1 + d2;
           return result;
   }

   private static String getInput(String prompt) {
           BufferedReader stdin = new BufferedReader(
                           new InputStreamReader(System.in));

           System.out.print(prompt);
           System.out.flush();

           try {
                   return stdin.readLine();
           } catch (Exception e) {
                   return "Error: " + e.getMessage();
           }
   }

}
```

Method Overloading (Multiple methods with same name/diff args)

Code:

```
public class Main {

   public static void main(String[] args) {

           int value1 = 5;
           int value2 = 10;
           int value3 = 15;
```

```
            int result = addValues(value1, value2, value3);
            System.out.println("The result is: " + result);

            String string1 = "10";
            String string2 = "25";
            int result2 = addValues(string1, string2);
            System.out.println("The result is: " + result2);
    }

    private static int addValues(int int1, int int2){
            return int1 + int2;
    }

    // will handle call if 3 values passed
    private static int addValues(int int1, int int2, int int3){
            return int1 + int2 + int3;
    }

    // handle different data types
    private static int addValues(String val1, String val2){
            int value1 = Integer.parseInt(val1);
            int value2 = Integer.parseInt(val2);
            return value1 + value2;
    }

}
```



Passing By Copy; Primitives

When calling a function, a COPY of the argument is passed to the method

```
            void incrementValue(int inFunction) {
                    inFunction ++;
                    System.out.println("In function: " + inFunction);
            }

            int original = 10;
            System.out.println("Original before : " + original);
            incrementValue(original);
            System.out.println("Original after : " + original);

            // Original before: 10
            // In function: 11
            // Original after: 10
    }
```

Passing by Reference: Complex Objects

- Variable of Complex Object is a Reference to Memory
- A Copy of the Complex Object is passed, but references the same memory addresses

```
inf[] original = {10,20,30};
```
original[0] > {10,20,30} < inFunction[0]

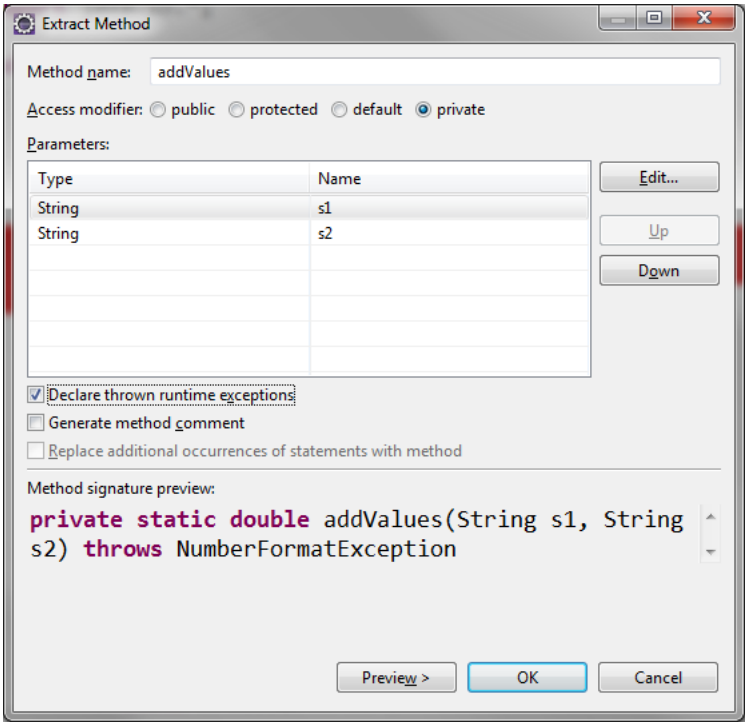
Passing Strings: Is Complex Object but Acts Like Primitive

```
void changeString(String inFunction) {
        inFunction = "New!";
        System.out.println("In function: " + inFunction);
}
```

- Strings are immutable, can't change after declaration
- Copy of entire string passed to function

```
            // Original before: Original!
            // In function: New!
            // Original after: Original!
```

Extract Method with Error Handling

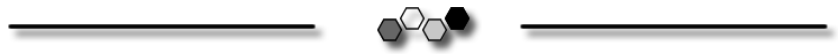


More Complex Programs and An Error Handling Declaration

Code:

```
import java.io.*;

public class Calculator2 {

    public static void main(String[] args) {
            String s1 = getInput("Enter a numeric value: ");
            String s2 = getInput("Enter a numeric value: ");
            String op = getInput("Enter 1=Add, 2=Subtract, 3=Multiply, 4=Divide: ");

            int opInt = Integer.parseInt(op);
            // declare variables BEFORE switch
            double result = 0;

            switch (opInt) {
            case 1:
                    result = addValues(s1, s2);
                    break;
            case 2:
                    result = subtractValues(s1, s2);
                    break;
            case 3:
                    result = multiplyValues(s1, s2);
                    break;
            case 4:
                    result = divideValues(s1, s2);
                    break;
            default:
                    System.out.println("Unknown Operation: " + op);
                    break;
            }

            System.out.println("The answer is " + result);
    }

    private static double divideValues(String s1, String s2) {
            double d1 = Double.parseDouble(s1);
            double d2 = Double.parseDouble(s2);
            double result = d1 / d2;
            return result;
    }

    private static double multiplyValues(String s1, String s2) {
            double d1 = Double.parseDouble(s1);
            double d2 = Double.parseDouble(s2);
            double result = d1 * d2;
            return result;
    }

    private static double subtractValues(String s1, String s2) {
            double d1 = Double.parseDouble(s1);
            double d2 = Double.parseDouble(s2);
            double result = d1 - d2;
            return result;
```

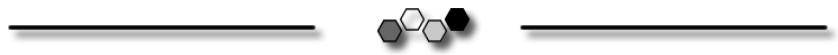
```
    }

    private static double addValues(String s1, String s2)
                    throws NumberFormatException {
            double d1 = Double.parseDouble(s1);
            double d2 = Double.parseDouble(s2);
            double result = d1 + d2;
            return result;
    }

    private static String getInput(String prompt) {
            BufferedReader stdin = new BufferedReader(
                            new InputStreamReader(System.in));

            System.out.print(prompt);
            System.out.flush();

            try {
                    return stdin.readLine();
            } catch (Exception e) {
                    return "Error: " + e.getMessage();
            }
    }

}
```



Using Equality Operators

Code:

```
public class Main {

    public static void main(String[] args) {
            String s1 = ("Welcome to California");
            String s2 = new String("Welcome to California");
            System.out.println(s2);

            if (s1 == s2) { //comparing OBJECTS not VALUES
                    System.out.println("With == They match");
            }
            else {
                    System.out.println("With == No match");
            }

            if (s1.equals(s2)) { //case sensitive
                    System.out.println("With .equals() They match");
            } else {
                    System.out.println("With .equals() No match");
            }

            // Prints string out one char at a time
            char[] chars = s1.toCharArray();
            for (char c : chars) {
                    System.out.println(c);
            }
    }

}
```

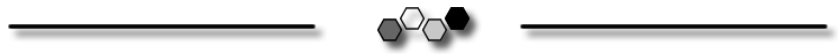


String Builder vs String Buffer

- Builder - single thread
- Buffer - multi threaded

Code:

```
public class Main {

    public static void main(String[] args) {
            String s1 = "Welcome";
            StringBuilder sb = new StringBuilder(s1);

            sb.append(" to California");
            System.out.println(sb);
            //string builder vs string buffer

    }

}
```



String Manipulation

Code:
```
public class Main {

    public static void main(String[] args) {

            String s1 = ("Welcome to California!");
            System.out.println("Length of string: " + s1.length());

            //String position
            int pos = s1.indexOf("California");
            System.out.println("Position of California: " + pos);

            // SubStrings
            String sub = s1.substring(11);
            System.out.println(sub);

            //
            String s2 = "Welcome!          ";
            int len1 = s2.length();
            System.out.println(len1);
            String s3 = s2.trim();
            System.out.println(s3.length());



    }

}
```

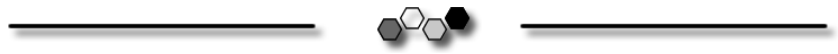
Date Manipulation

Code:
```
import java.text.DateFormat;
import java.util.Date;
import java.util.GregorianCalendar;


public class Main {

    public static void main(String[] args) {
            Date d = new Date();
            System.out.println(d);
            //Fri Aug 03 17:43:06 CDT 2012

            //Date class that can add days to, etc
            GregorianCalendar gc = new GregorianCalendar(2009, 1, 28);
            System.out.println(gc);

//java.util.GregorianCalendar[time=1235887200000,areFieldsSet=true,areAllFieldsSet=true,lenient=true,zone=sun.util.calen
            //need to prep for printing....

            //add gc field (YEAR/MONTH/DATE), add by how much
            gc.add(GregorianCalendar.DATE, 1);

            //prep for printing
            Date d2 = gc.getTime();
            System.out.println(d2);
            //Sun Mar 01 00:00:00 CST 2009

            //factoring method, returns an instance of that class
            DateFormat df = DateFormat.getDateInstance();
            String sd = df.format(d2);
            System.out.println(sd);
            //Mar 1, 2009

            // can format the Date object in many ways..
            DateFormat df2 = DateFormat.getDateInstance(DateFormat.FULL);
            String sd2 = df2.format(d2);
            System.out.println(sd2);
            //Sunday, March 1, 2009

    }

}
```

Error Handling

Code:
```
public class Main {

    public static void main(String[] args) {

            //declarations
            //String s;...need to assign value
            String s = null;
            System.out.println(s);

            //Arrays
            String[] strings = {"Welcome!"}; //one item in array
```

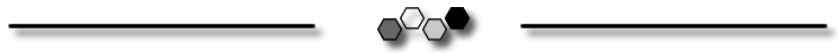
```
            System.out.println(strings[1]);  // error, no [1], only [0]
            //Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1
            //at Main.main(Main.java:13)
    }
}
```



Exception Handling

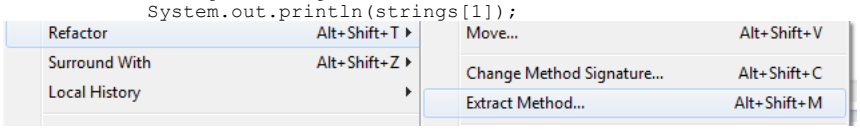
Code:

```
public class Main {

    public static void main(String[] args) {
            //try catch block

            try {
                    String[] strings = {"Welcome!"};
                    System.out.println(strings[1]);
            } catch (ArrayIndexOutOfBoundsException e) {
                    //e.printStackTrace();
                    System.out.println("Error occurred");
            }
            // without try/catch block
            // Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1
            // at Main.main(Main.java:6)

            //with try/catch blcok
            //The application is still running!
            //java.lang.ArrayIndexOutOfBoundsException: 1
            //at Main.main(Main.java:9)

            //with error handling
            //Error occurred
            //The application is still running!

            System.out.println("The application is still running!");


    }

}
```

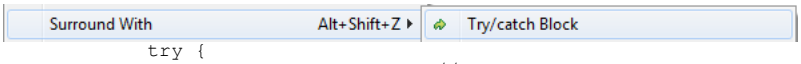


Extracting and Error Handling with Try/Catch Block: Revisited

```
            String[] strings = {"Welcome!"};
            System.out.println(strings[1]);
```




```
..
                    getArrayItem(); //refactored getArrayItem()
..
        private static void getArrayItem()
            throws ArrayIndexOutOfBoundsException {
            String[] strings = {"Welcome!"};
            System.out.println(strings[1]);
    }
```

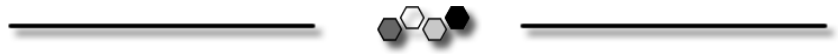



```
            try {
                    getArrayItem(); //refactored getArrayItem()
            } catch (ArrayIndexOutOfBoundsException e) {
                    // e.printStackTrace(); // <--- throws ugly message
                    System.out.println("Array item was out of bounds");
            }
```

Code:

```
public class Main {

    public static void main(String[] args) {
            //Extrac Method with Error Handling
            //Surround with try/catch blcok
            try {
                    getArrayItem(); //refactored getArrayItem()
            } catch (ArrayIndexOutOfBoundsException e) {
                    // e.printStackTrace(); // <--- throws ugly message
                    System.out.println("Array item was out of bounds");
                    //Array item was out of bounds
            }
    }

    private static void getArrayItem()
            throws ArrayIndexOutOfBoundsException {
            String[] strings = {"Welcome!"};
            System.out.println(strings[1]);
    }
```

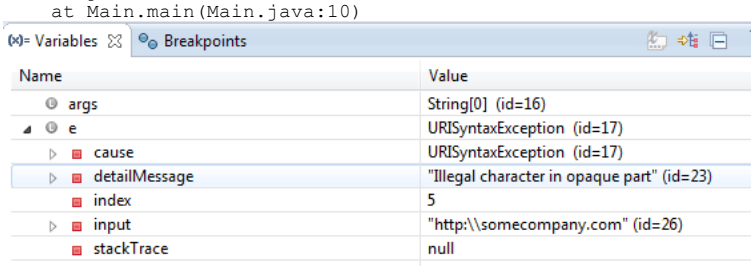
```
}
```



Debugger

Finding Possible Exceptions
Highlight command > Help > Dynamic Help > JavaDoc > CONSTRUCTOR > METHOD

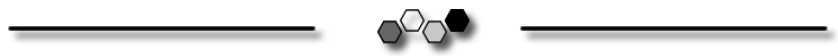
```
 URI uri = new URI("http:\\somecompany.com");

java.net.URISyntaxException: Illegal character in opaque part at index 5:
http:\somecompany.com
    at java.net.URI$Parser.fail(Unknown Source)
    at java.net.URI$Parser.checkChars(Unknown Source)
    at java.net.URI$Parser.parse(Unknown Source)
    at java.net.URI.<init>(Unknown Source)
    at Main.main(Main.java:10)
```




```
From        e.printStackTrace();
To        System.out.println(e.getMessage());
```

Code:

```
import java.net.URI;
import java.net.URISyntaxException;


public class Main {

    public static void main(String[] args) {
            //Uniform Resource Identifier
            try {
                    URI uri = new URI("http:\\somecompany.com");
            } catch (URISyntaxException e) {
                    System.out.println(e.getMessage());
                    /*
                    e.printStackTrace();
                    java.net.URISyntaxException: Illegal character in opaque part at
index 5: http:\somecompany.com
                    at java.net.URI$Parser.fail(Unknown Source)
                    at java.net.URI$Parser.checkChars(Unknown Source)
                    at java.net.URI$Parser.parse(Unknown Source)
                    at java.net.URI.<init>(Unknown Source)
                    at Main.main(Main.java:10)
                    */
            }

            System.out.println("I'm alive!");
            //Exception in thread "main" java.lang.Error: Unresolved compilation
problem:
            //    Unhandled exception type URISyntaxException
            //    at Main.main(Main.java:8)

    }

}
```



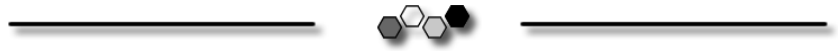
Simple Arrays

Arrays NOT resizeable at runtime..

Code:

```
public class Main {

    public static void main(String[] args) {

            int[] a1 = new int[3]; //array of 3 integers
            for (int i = 0; i < a1.length; i++) {
                    System.out.println(a1[i]);
            }
```

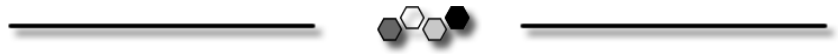
```
            int a2[] = new int[3]; //array of 3 integers
            for (int i = 0; i < a2.length; i++) {
                    System.out.println(a2[i]);
            }

            int[] a3 = {3,6,9};
            for (int i = 0; i < a3.length; i++) {
                    System.out.println(a3[i]);
            }

            System.out.println("The value of the first item is: " + a3[0]);

    }
}
```



2 Dimensional Arrays

Code:

```
public class Main {

    public static void main(String[] args) {

            String[][] states = new String[3][2];
            states[0][0] = "California";
            states[0][1] = "Sacramento";
            states[1][0] = "Oregon";
            states[1][1] = "Salem";
            states[2][0] = "Washington";
            states[2][1] = "Olympia";

            for (int i = 0; i < states.length; i++) {
                    StringBuilder sb = new StringBuilder();
                    for (int j = 0; j < states[i].length; j++) {
                            if (j == 0) {
                                    sb.append("The Capitol of ");
                            } else {
                                    sb.append(" is ");
                            }
                            sb.append(states[i][j]);
                    }
            System.out.println(sb);
            }
    }

}
```



MultiDimensional Arrays

Code:

```
import java.util.ArrayList;


public class Main {

    public static void main(String[] args) {

            //Diamond Operator <E> = Specific DataType
            ArrayList<String> list = new ArrayList<String>();

            list.add("California");
            list.add("Oregon");
            list.add("Washington");

            System.out.println(list);

            // Allows changing array size at runtime
            list.add("Alaska");
            System.out.println(list);

            list.remove(0);
            System.out.println(list);

            String state = list.get(1);
            System.out.println("The second state is " + state);

            int pos = list.indexOf("Alaska");
            System.out.println("Alaska is at position " + pos);

            Boolean b = list.contains("California");
            System.out.println(b);
            b = list.contains("Alaska");
            System.out.println(b);
    }
}
```

HashMap: Unordered data collections

Code:
import java.util.HashMap;
// HashMap is for storing unordered data collections

```
public class Main {

    public static void main(String[] args) {
            // <these are> generics
            HashMap<String, String> map = new HashMap<String, String>();
            map.put("California", "Sacramento");
            map.put("Oregon", "Salem");
            map.put("Washington", "Olympia");

            System.out.println(map);
            //{California=Sacramento, Oregon=Salem, Washington=Olympia}

            map.put("Alaska", "Juneau");
            System.out.println(map);
            //{California=Sacramento, Oregon=Salem, Washington=Olympia,
Alaska=Juneau}

            String cap = map.get("Oregon");
            System.out.println("The capitol of Oregon is " + cap);
            //The capitol of Oregon is Salem

            map.remove("California");
            System.out.println(map);
            //{Oregon=Salem, Washington=Olympia, Alaska=Juneau}

    }

}
```

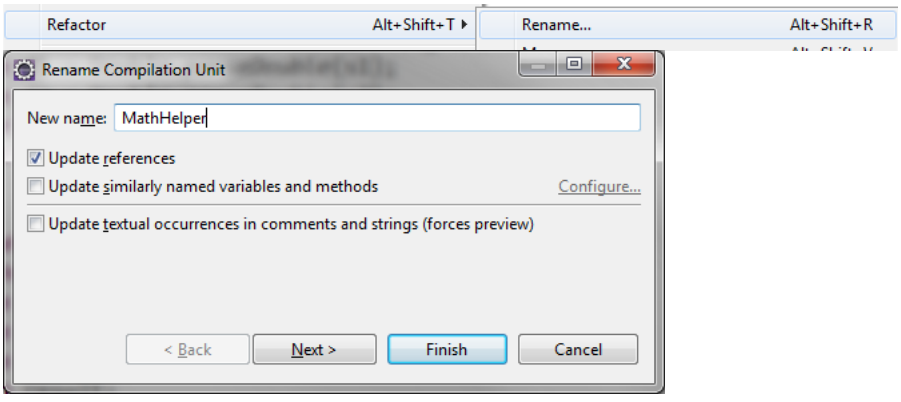
Encapsulation

- Encapsulate functions by wrapping sections of code in a class/method
- Now reuseable
- Changing in one place, not many

Code:

Classes

- Class = Filename.java
- Only one public class per java file
- Multiple classes only accessible within the files
- Refactoring is process of pulling code out, creating method/class
    - Also changing class name is refactoring



Code:

From

```
result = divideValues(s1, s2);
```

To

```
result = SimpleMath.divideValues(s1, s2);
```

From

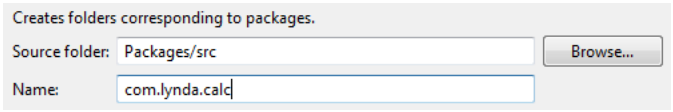
```
---------------------------------------------------------
public class Calculator2 {

   private static double divideValues(String s1, String s2) {
         double d1 = Double.parseDouble(s1);
         double d2 = Double.parseDouble(s2);
         double result = d1 / d2;
         return result;
---------------------------------------------------------
```

To

```
---------------------------------------------------------
public class SimpleMath {

   public static void main(String[] args) {


   }

   public static double divideValues(String s1, String s2) {
         double d1 = Double.parseDouble(s1);
         double d2 = Double.parseDouble(s2);
         double result = d1 / d2;
         return result;
---------------------------------------------------------
```

Packages

- If class is anywhere but default package, it must be declared
- when creating packages, use reverse domain...
    - com.silosix.calc



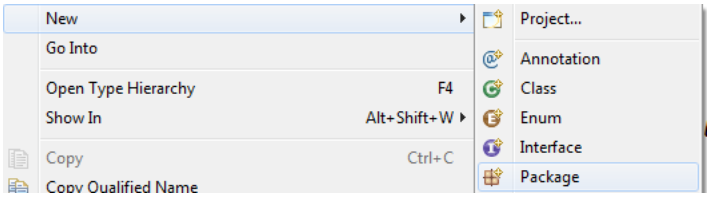
Importing Packages

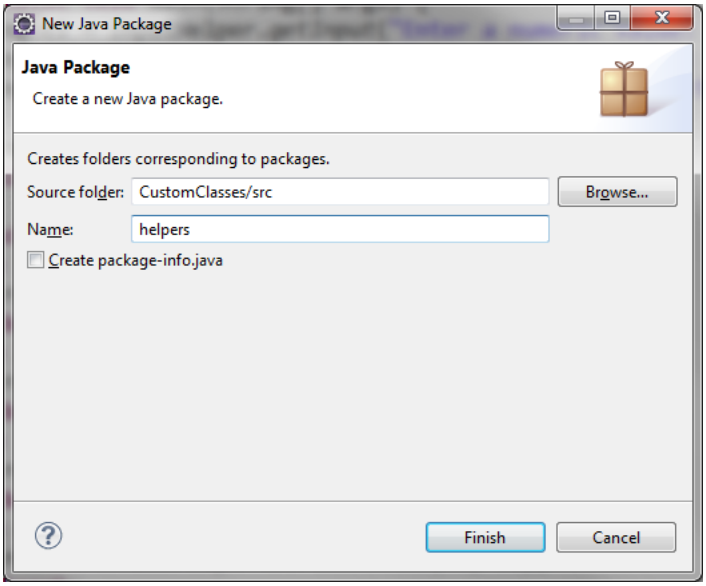
```
package com.lynda.calc;
import com.lynda.calc.helpers.InputHelper;
import com.lynda.calc.helpers.MathHelper;
```

same as:

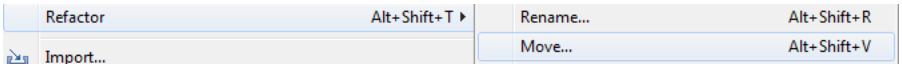
```
package com.lynda.calc;
import com.lynda.calc.helpers.*;
```

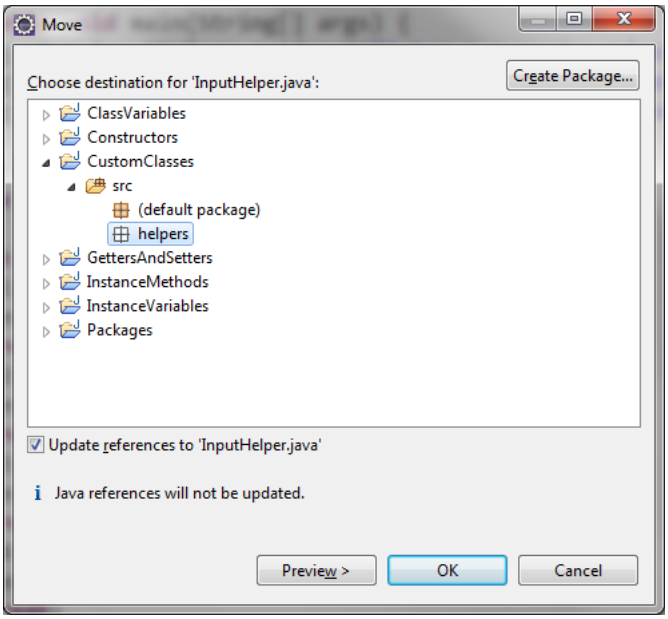
CTRL-O will change * to specific class imports

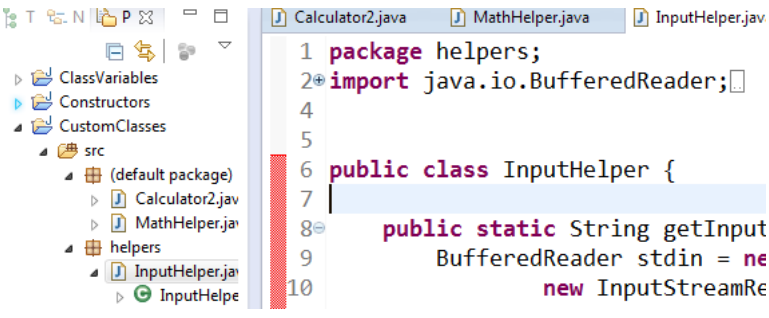




Refactor > Move





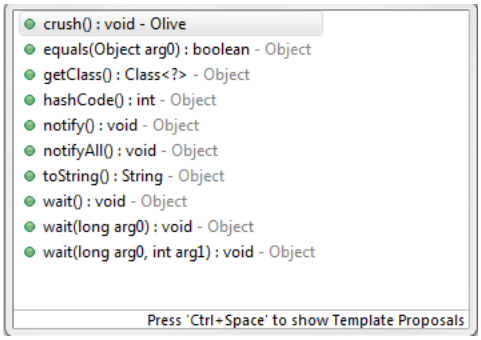


Instance Methods

- Class method called from definition of the class
  - building up utility functions that pass all data in the call
  - STATIC present

- Instance method call from instance of the class - OBJECT
  - objects stick around and retain their data so its always accessible.
  - STATIC missing
- Method declarations
  - static - class method
  - public - called anywher ein ap
  - private - only within class
  - protected - only within this class or its subclasses

- Object Superclass methods with
- Olive()'s crush() method
- Olive inherits Object() methods/properties



Code: Main.java
```
package com.lynda.olivepress;

import com.lynda.olivepress.olives.Olive;
import com.lynda.olivepress.press.OlivePress;

public class Main {
    public static void main(String[] args) {
            //creating 3 anonymous Olive objects
            Olive[] olives = {new Olive(), new Olive(), new Olive()};
            OlivePress press = new OlivePress();
            press.getOil(olives);
    }
}
```

Code: OlivePress.java
```
package com.lynda.olivepress.press;

import com.lynda.olivepress.olives.Olive;

public class OlivePress {
    public void getOil(Olive[] olives)    {
            for (Olive olive : olives) {
                    olive.crush();
            }
    }
}
```

Code: Olive.java
```
package com.lynda.olivepress.olives;

public class Olive {

    public void crush() {
            System.out.println("Ouch!");
    }
}
```

Instance Variables (Not Static)

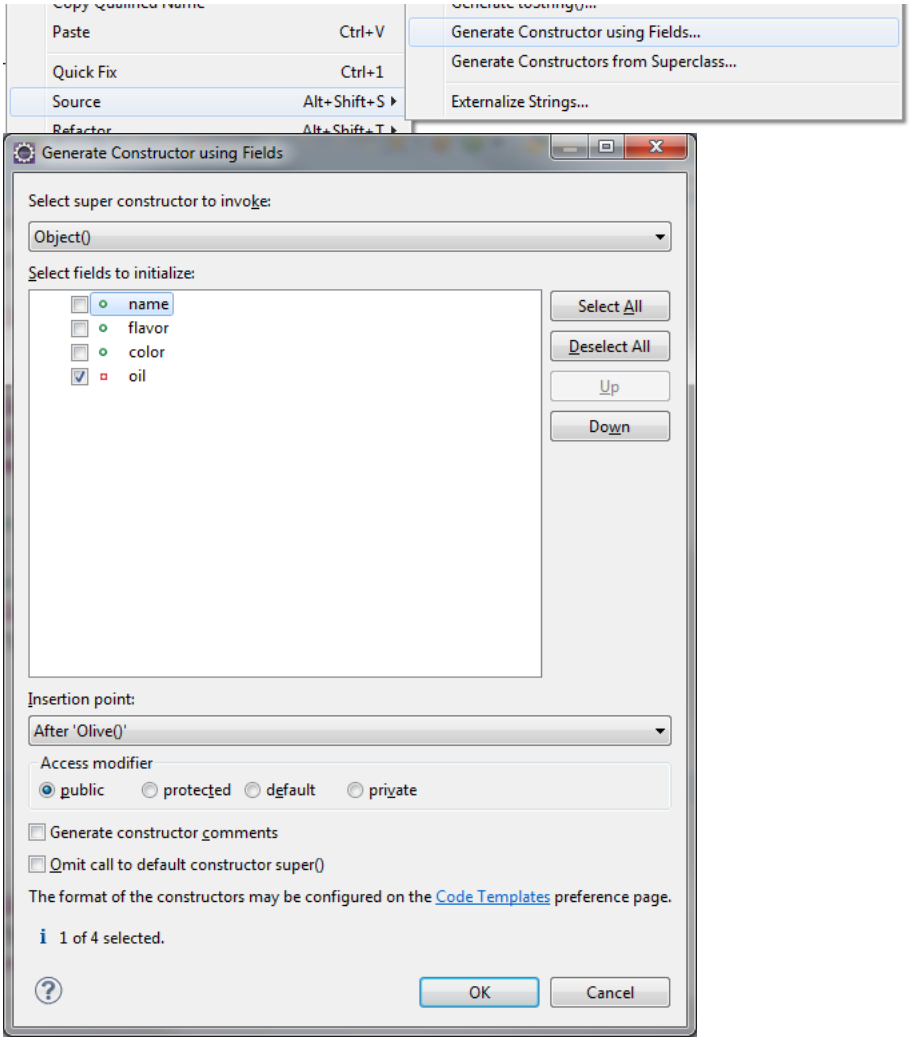
Constructors

- Constructors have no return value (void, int, etc)
- can create multiple constructors (overloading) with different input specs
- Always create a 'no argument' constructor for clarity
  - public OlivePress() { }
- Can Create a new constructor with fields....

Constructor of the Olive() class
```
    public Olive() {
        System.out.println("Constructor of " + this.name);
    }
```

Creating another constructor to catch argument and populate a field:

```
        public Olive(int oil) {
//this.oil means field(instance variable
//otherwise refers to argument
            this.oil = oil;

      }
```



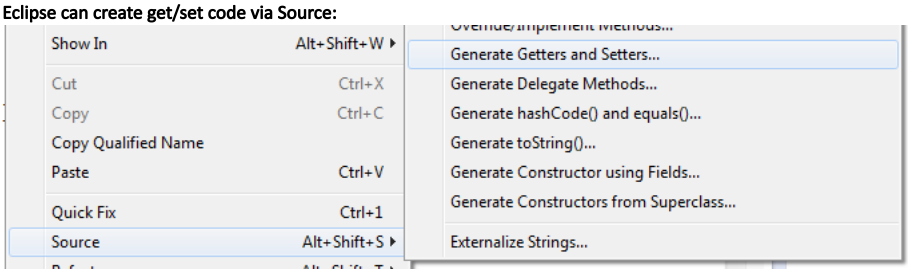
Code:

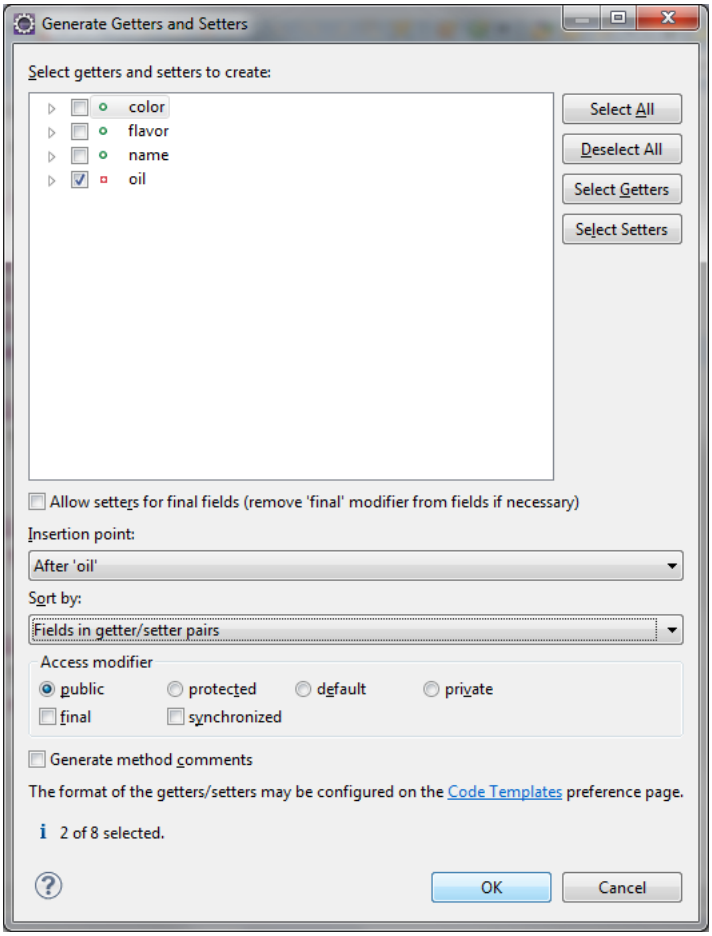
```
package com.lynda.olivepress.olives;

public class Olive {

    public String name = "Kalamata";
    public String flavor = "Grassy";
    public long color = 0x000000;
    private int oil = 3;

    //constructor, same name as class
    //no return on constructors
    //can overload the constructor
    public Olive() {
            System.out.println("Constructor of " + this.name);
    }

    public Olive(int oil) {
            this.oil = oil;
    }

    public int crush() {
            System.out.println("ouch!");
            return oil;
    }
}
```

Getters/Setters

- OO development patters
- Fields should be private
- Get to data with get/set
    - Create private get() and set()

Eclipse can create get/set code via Source:





Creates:

```
        public int getOil() {
            return oil;
    }

    public void setOil(int oil) {
            this.oil = oil;
    }
```

Code: Main.java

```
package com.lynda.olivepress;

import java.util.ArrayList;

import com.lynda.olivepress.olives.Olive;
import com.lynda.olivepress.press.OlivePress;

public class Main {

    public static void main(String[] args) {

            ArrayList<Olive> olives = new ArrayList<Olive>();

            Olive olive;

            olive = new Olive(2);
            System.out.println(olive.name);
```

```
            olives.add(olive);

            olive = new Olive(1);
            System.out.println(olive.name);
            olives.add(olive);

            olive = new Olive(2);
            System.out.println(olive.name);
            olives.add(olive);

            OlivePress press = new OlivePress();
            press.getOil(olives);
            System.out.println("You got " + press.getTotalOil() + " units of oil");

            press.getOil(olives);
            System.out.println("You got " + press.getTotalOil() + " units of oil");
    }

}
```

Code: OlivePress.java

```
package com.lynda.olivepress;

import java.util.ArrayList;

import com.lynda.olivepress.olives.Olive;
import com.lynda.olivepress.press.OlivePress;

public class Main {

    public static void main(String[] args) {

            ArrayList<Olive> olives = new ArrayList<Olive>();

            Olive olive;

            olive = new Olive(2);
            System.out.println(olive.name);
            olives.add(olive);

            olive = new Olive(1);
            System.out.println(olive.name);
            olives.add(olive);

            olive = new Olive(2);
            System.out.println(olive.name);
            olives.add(olive);

            OlivePress press = new OlivePress();
            press.getOil(olives);
            System.out.println("You got " + press.getTotalOil() + " units of oil");

            press.getOil(olives);
            System.out.println("You got " + press.getTotalOil() + " units of oil");
    }

}
```

Code: Olive.java

```
package com.lynda.olivepress;

import java.util.ArrayList;

import com.lynda.olivepress.olives.Olive;
import com.lynda.olivepress.press.OlivePress;

public class Main {

    public static void main(String[] args) {

            ArrayList<Olive> olives = new ArrayList<Olive>();

            Olive olive;

            olive = new Olive(2);
            System.out.println(olive.name);
            olives.add(olive);

            olive = new Olive(1);
            System.out.println(olive.name);
            olives.add(olive);

            olive = new Olive(2);
            System.out.println(olive.name);
            olives.add(olive);

            OlivePress press = new OlivePress();
            press.getOil(olives);
            System.out.println("You got " + press.getTotalOil() + " units of oil");

            press.getOil(olives);
            System.out.println("You got " + press.getTotalOil() + " units of oil");
    }

}
```

Class Variables

- Java has no CONSTANT declaration so.....

```
// public - accessible from entire app
// static - class var
// final - value can't be changed
```

IN Olive()...

```
public static final long BLACK= 0x000000;
```

using it

```
public long color = Olive.BLACK;
```

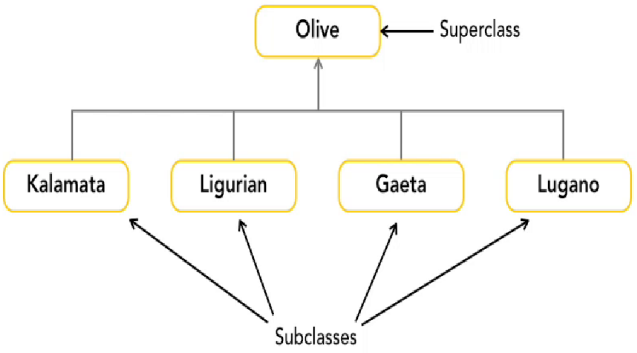
Inheritance

- Java has only single inheritance - only one inherited parent
- Parent/child
- Base/derived
- Superclass/subclass <- Preferred Java nomenclature
    - By default Object() is the superclass unless directly specified

Polymorphism

- Can used as Superclass or Subclass
- Declare the object by Superclass

Superclass can have more than one subclass



- **Private - only called within own class**
- **Protected - called by own class or subclass**
- **Public - called from anywhere**

Subclasses extend superclass

extending Olive by setting inital volume (setVolume)

```
public class Kalamata extends Olive() {
        public Kalamata() {this.setVolume(2);}
}
public class Liguria extends Olive() {
        public Liguria () {this.setVolume(5);}
}
```

.....this creates inheritance

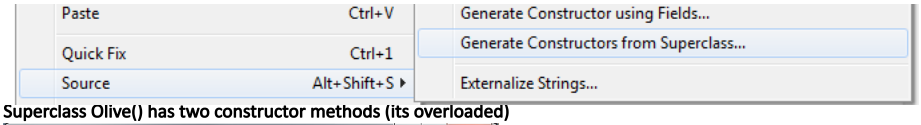
```
Olive[] olives = {new Kalamata(), new Liguria(), new Kalamata()};
OlivePress press = new OlivePress(olives);
OliveOil oil - press.getOil;
```

Takes Kalamata() class and fits into Superclass Olive()

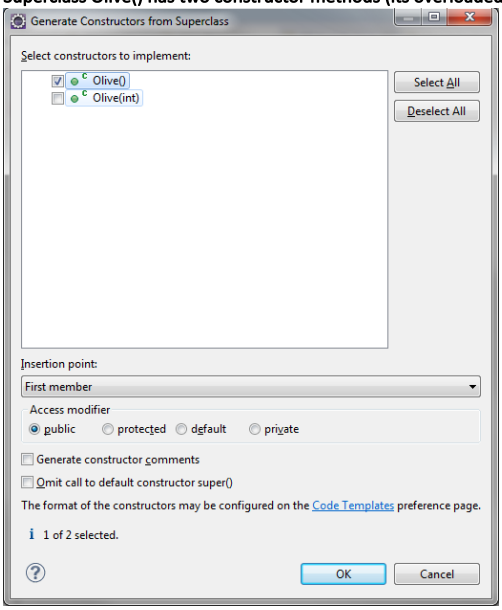
Extending Custom Classes

- Superclass doesnot pass on its constructor so...
- Each subclass needs its own constructor

In subclass... use IDE to copy constructors from the Superclass
Can select all or one...



Superclass Olive() has two constructor methods (its overloaded)



Creates...

```
        public Kalamata() {
            super();//calling superclass constructor method
            // TODO Auto-generated constructor stub
    }
```

Code: Main

```
package com.lynda.olivepress;

import java.util.ArrayList;

import com.lynda.olivepress.olives.Kalamata;
import com.lynda.olivepress.olives.Ligurian;
import com.lynda.olivepress.olives.Olive;
import com.lynda.olivepress.press.OlivePress;

public class Main {

    public static void main(String[] args) {

            ArrayList<Olive> olives = new ArrayList<Olive>();

            Olive olive;

            //olive = new Olive(2); //Was calling SuperClass
            olive = new Kalamata();
            System.out.println(olive.name);
            olives.add(olive);

            olive = new Ligurian();
            System.out.println(olive.name);
            olives.add(olive);

            olive = new Kalamata();
            System.out.println(olive.name);
            olives.add(olive);

            OlivePress press = new OlivePress();
            press.getOil(olives);

            System.out.println("You got " + press.getTotalOil() +
                            " units of oil");

            press.getOil(olives);

            System.out.println("You got " + press.getTotalOil() +
                            " units of oil");
    }
}
```

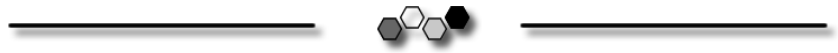
Code:Olive.java

```
package com.lynda.olivepress.olives;

public class Olive {

    public static final long BLACK = 0x000000;
    public static final long GREEN = 0x00ff00;
```

```
    public String name = "Kalamata";
    public String flavor = "Grassy";
    public long color = Olive.BLACK;
    private int oil = 3;

    public int getOil() {
            return oil;
    }

    public void setOil(int oil) {
            this.oil = oil;
    }

    public Olive() {
            System.out.println("Constructor of " + this.name);
    }

    public Olive(int oil) {
            setOil(oil);
    }

    public int crush() {
            System.out.println("ouch!");
            return oil;
    }
}
```

Code:Kalamata

```
package com.lynda.olivepress.olives;

public class Kalamata extends Olive {

    public Kalamata() {
            super(2); //calling superclass constructor method and passing '2'
            this.name = "Kalamata";
            this.flavor = "Grassy";
            this.color = Olive.BLACK;
    }

}
```



Overriding Methods (super.something();)

Code: Olive.java

```
package com.lynda.olivepress.olives;

public class Olive {

    public static final long BLACK = 0x000000;
    public static final long GREEN = 0x00FF00;

    public String name = "Kalamata";
    public String flavor = "Grassy";
    public long color = Olive.BLACK;
    private int oil = 3;

    public int getOil() {
            return oil;
    }

    public void setOil(int oil) {
            this.oil = oil;
    }

    public Olive() {
            System.out.println("Constructor of " + this.name);
    }

    public Olive(int oil) {
            setOil(oil);
    }

    public int crush() {
            System.out.println("crush from superclass");
            //System.out.println("ouch!");
            return oil;
    }

}
```

Code: Kalamata.java

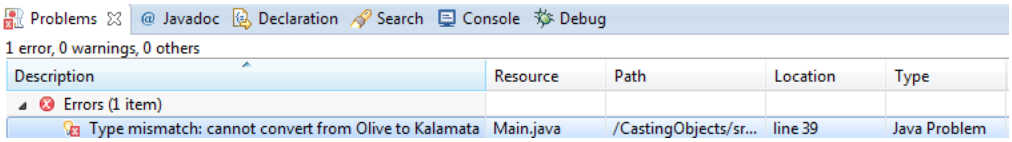
```
package com.lynda.olivepress.olives;

public class Kalamata extends Olive {

    public Kalamata() {
            super(2);
            this.name = "Kalamata";
            this.flavor = "Grassy";
            this.color = Olive.BLACK;
    }
```

```
    //Annotation wth @...data type MUST match (super.crush())
    @Override
    public int crush() {
            System.out.println("crush from subclass");
            return super.crush();
    }

}
```

Casting Objects

- As in conversion...upward/downward (int -> long / long -> int)
- casting
  - upcasting - subclass as superclass (SAFE)
  - downcasting - superclass as subclass (RISKY)

```
//Downcasting - will cause compiler error
Kalamata olive1 =olives.get(0);

//Downcasting Excplicitly
Kalamata olive1 = (Kalamata)olives.get(0);
```



Create a Kalamata() olive1 from Olive() in ArrayList[0], position 0

```
Kalamata olive1 = olives.get(0);
```

Create a Kalamata() olive1 from Kalamata() Olive() in ArrayList[0], position 0

```
Kalamata olive1 = (Kalamata)olives.get(0);
```

Code:

Main.java

```
        //Downcasting
        Kalamata olive1 = (Kalamata)olives.get(0);
    //downcast Olive() to (Kalamata)
    //(Kalamata)olives.get(0) means USE SUBCLASS
        System.out.println("Olive 1 is from " + olive1.getOrigin());
```

Kalamata.java (only in the Kalamata subclass, not others..)
getOrigin extends Olive()

```
    public String getOrigin() {
        return "Greece";
}
```

Interfaces

- Allows definition of classes' structure
  - final fields
  - method names
  - return data type
- Interfaces provides definition for creating classes
  - Allows for polymorphism due to similarities

Code:

This method only accept ArrayLists (part of Collection data type I/F)

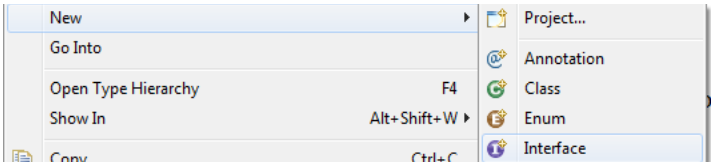
```
public void getOil(ArrayList<Olive> olives) {}
```

This is more flexible and can take any Data Type that implements the Collection I/F

```
public void getOil(Collection<Olive> olives) {
```

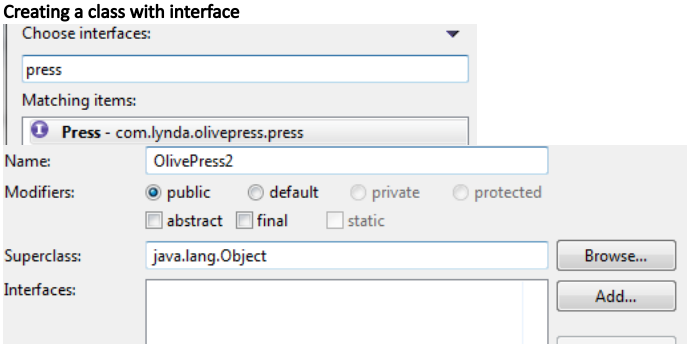
Creating Interfaces

- No constructor methods or other class elements
- Modeling behavior, not dynamic managment of data
- MUST BE PUBLIC



```
package com.lynda.olivepress.press;

public interface Press {
}
```

Creating a class with interface



Code:

```
package com.lynda.olivepress.press;

import java.util.Collection;

import com.lynda.olivepress.olives.Olive;

public class OlivePress2 implements Press {

    @Override
    public void getOil(Collection<Olive> olives) {
            // TODO Auto-generated method stub

    }

    @Override
    public int getTotalOil() {
            // TODO Auto-generated method stub
            return 0;
    }

    @Override
    public void setTotalOil(int totalOil) {
            // TODO Auto-generated method stub
    }
}
```

File I/O: Copy a Text File

Code:

```
package com.lynda.files;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class CopyFile {

    public static void main(String[] args) {
            try {
                    // file is in the PROJECT directory
                    File f1 = new File("loremipsum.txt");
                    File f2 = new File("target.txt");

                    InputStream in = new FileInputStream(f1);
                    OutputStream out = new FileOutputStream(f2);

                    //Copy text file byte by byte..or by chunck of bytes
                    byte[] buf = new byte[1024];
                    int len; //holds bytes remaining
                    //reading information to fill the array
                    //return the total number of bytes received to len
                    while ((len = in.read(buf)) > 0) {
                            out.write(buf, 0, len);
                    }

                    in.close();
                    out.close();

                    System.out.println("File copied");
            } catch (FileNotFoundException e) {
                    // TODO Auto-generated catch block
```

```
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

    }
}
```

ApacheFileUtils Library

- Most developers put jars in the project's \lib folder
- Add to Build path: Access Denied
  - Clear Hidden Attribute
  - Set Everyone for 'Full Control'

Code:

```
package com.lynda.files;

import java.io.BufferedInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;

public class ReadNetworkFile {

    public static void main(String[] args) {

            try {
                URL url = new
URL("http://services.explorecalifornia.org/rss/tours.php");
                InputStream stream = url.openStream();
                BufferedInputStream buf = new BufferedInputStream(stream);

                StringBuilder sb = new StringBuilder();

                while (true) {
                        //buff.read will return number of character read
                        //will send -1 if end of file/stream
                        int data = buf.read();

                        if (data == -1) {
                                break;
                        }
                        else {
                                //convert data to char
                                sb.append((char)data); //Type Casting
                        }
                }
                System.out.println(sb);
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
    }
}
```

ParseXML

- get jdom or use java base classes:

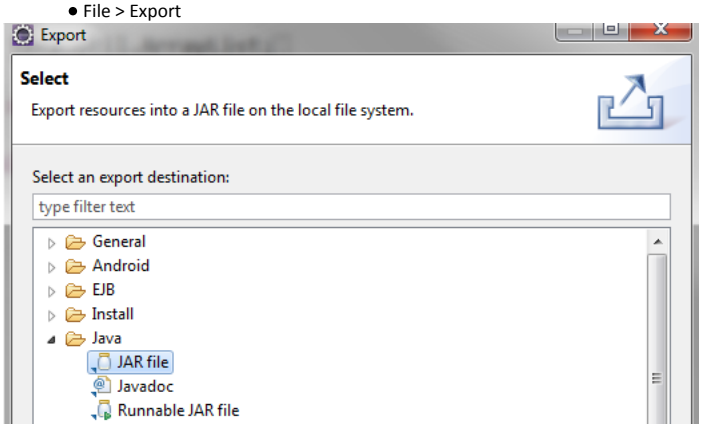
Code:

```
package com.lynda.files;

import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class ReadXML {

    public static void main(String[] args) {

            try {
                DocumentBuilderFactory factory =
                        DocumentBuilderFactory.newInstance();
                DocumentBuilder builder = factory.newDocumentBuilder();
```

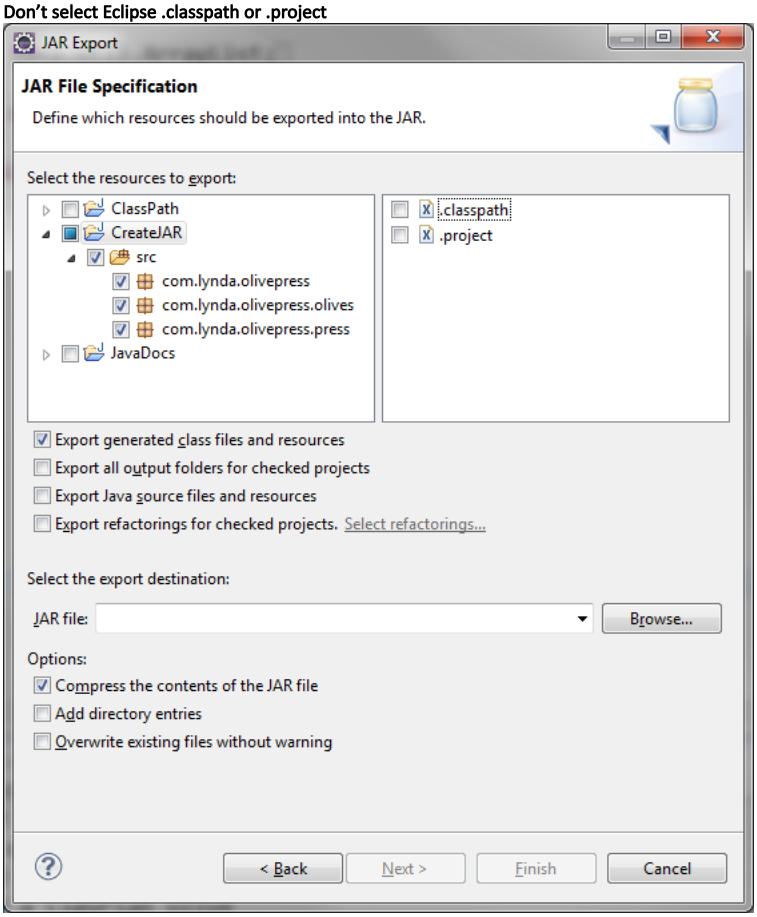
```
                    Document doc =
builder.parse("http://services.explorecalifornia.org/rss/tours.php");

                    NodeList list = doc.getElementsByTagName("title");
                    System.out.println("There are " + list.getLength() + " items");

                    for (int i = 0; i < list.getLength(); i++) {
                            Element item = (Element)list.item(i);
                            System.out.println(item.getFirstChild().getNodeValue());
                    }

            } catch (ParserConfigurationException e) {
                    e.printStackTrace();
            } catch (SAXException e) {
                    e.printStackTrace();
            } catch (IOException e) {
                    e.printStackTrace();
            }
    }
}
```

Creating JARs

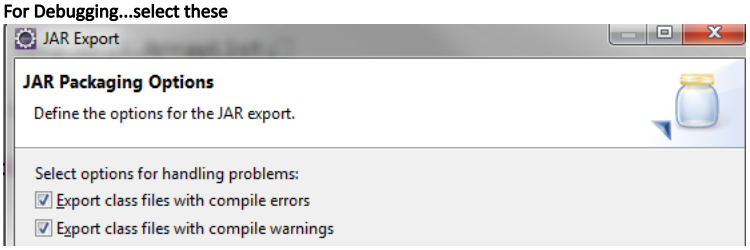
- Build Project
- File > Export



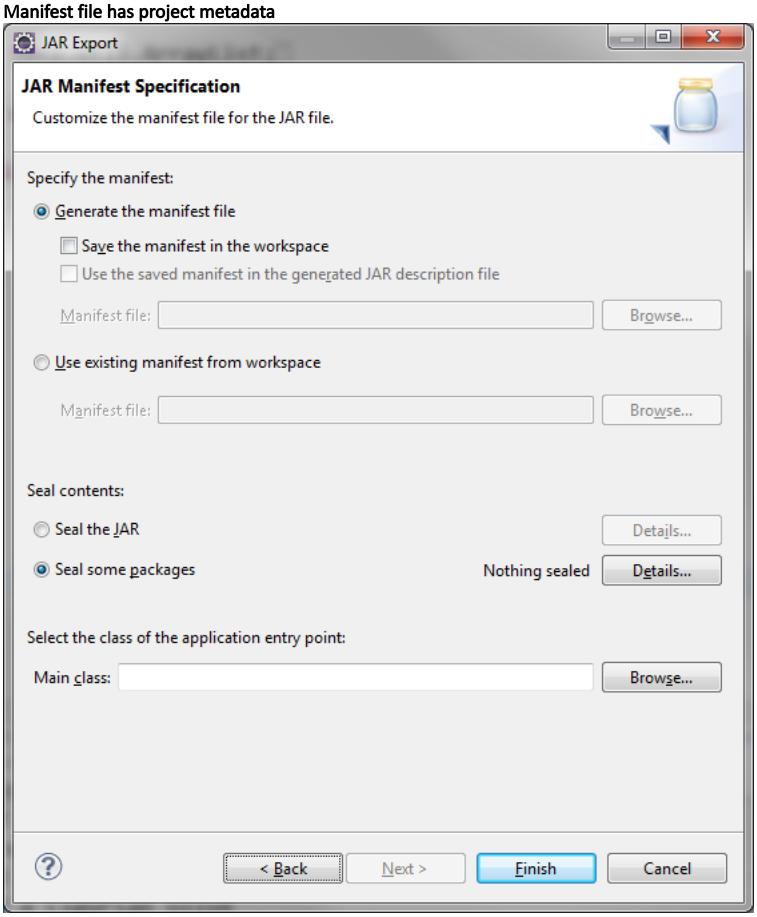
Don't select Eclipse .classpath or .project



For Debugging...select these



Manifest file has project metadata



ClassPath

Create a batch file and pass %1

```
set CLASSPATH=.
```

OR

For Linux

```
D:\TEMP\Eclipse>java -classpath .:OlivePressApp.jar com.lynda.olivepress.Main
```

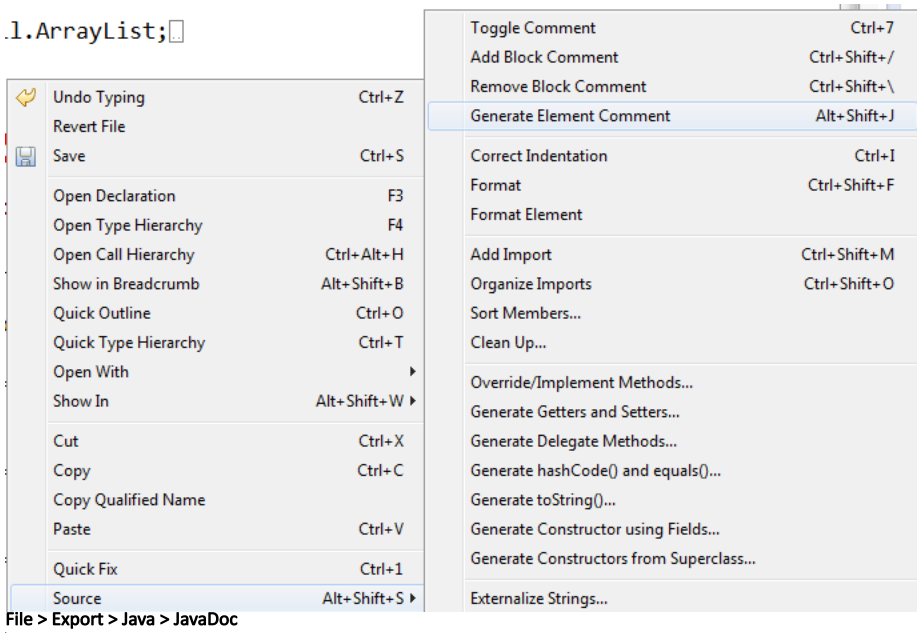
For Windows

```
D:\TEMP\Eclipse>java -classpath .;OlivePressApp.jar com.lynda.olivepress.Main
You crushed a Kalamata olive
You crushed a Ligurian olive
You crushed a Kalamata olive
You have 5 units of oil
You crushed a Kalamata olive
You crushed a Ligurian olive
You crushed a Kalamata olive
Now you have 10 units of oil
Olive 1 is from Greece
```

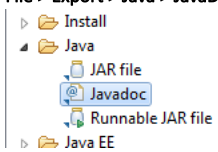
JavaDocs

Source > Generate Element Comment

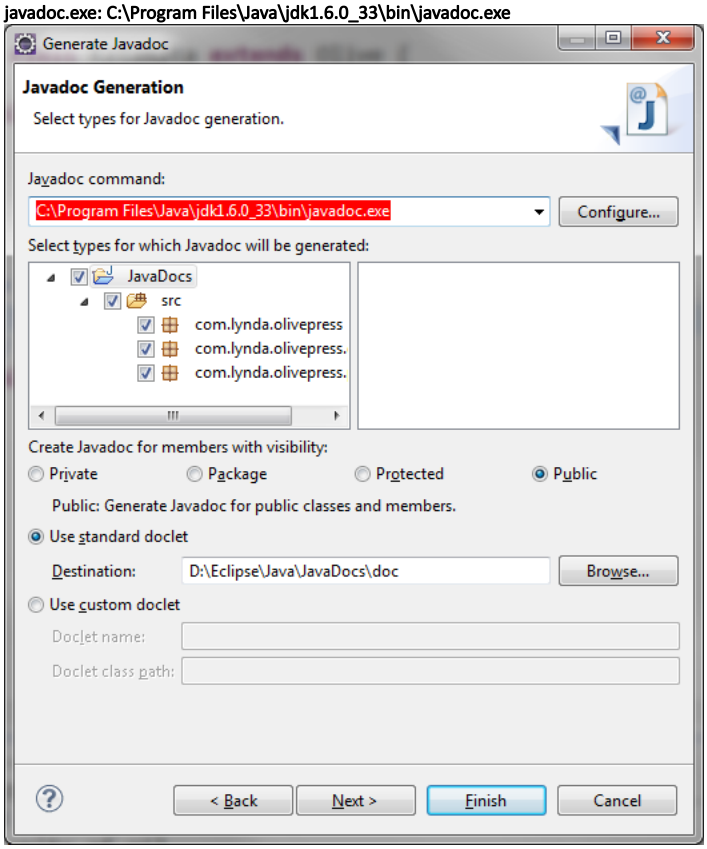
```
/**
 * @author SiloSix
 *
 */
```

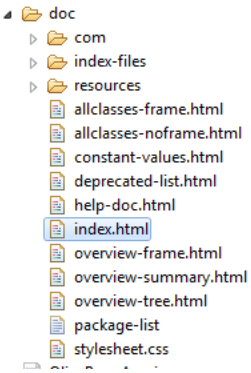


File > Export > Java > JavaDoc



javadoc.exe: C:\Program Files\Java\jdk1.6.0_33\bin\javadoc.exe





Resources

- apache commons

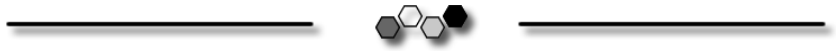
JUnit Class for testing:

- Annotations
  - @Test, @Before, @After, @BeforeClass, @AfterClass, @Ignore

Code:

```
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Ignore;
import org.junit.Test;


public class myJUnit1 {
   //No main() method, so JUnit will take over

   @BeforeClass
   public static void mBeforeTestClass(){
         System.out.println("--------ClassBegin----------");
   }
   //Annotation: Before EACH @Test
   @Before
   public void mBeforeTest(){
         System.out.println("--------------------------");
   }


   //Gets executed every time we run the JUnit program
   @Test
   public void test1(){
         if (mMultiply(10,30)==300) {
               System.out.println("Multiply Pass");
         } else {
               System.out.println("Multiply Fail");
               fail("Multiply Failed for 10 and 30");
         }
   }

   //Test 2 code
   @Test
   public void test2(){
         if (mAdd(10,30)==300){
               System.out.println("Add Pass");
         } else {
               System.out.println("Add Fail");
               fail("Add Failed for 10 and 30");
         }
   }

   //Test 3 code
   @Test
   public void test3(){
         if(mDivide(10,30)==300) {
               System.out.println("Divide Pass");
         } else {
               System.out.println("Divide Fail");
               fail("Divide Failed for 10 and 30");
         }
   }


   //Test 4 code won't run due to @Ignore
   @Ignore
   @Test
   public void test4(){
```

```
        if(mDivide(10,30)==300) {
            System.out.println("Divide Pass");
        } else {
            System.out.println("Divide Fail");
            fail("Divide Failed for 10 and 30");
        }
    }

    // Runs after EACH @Test
    @After
    public void mAfterTest(){
        System.out.println("---------------------------");
    }


    @AfterClass
    public static void mAfterTestClass(){
        System.out.println("---------Class End----------------");
    }
    //Multiply
    public int mMultiply(int x, int y){
        return x*y;
    }

    //Add
    public int mAdd(int x, int y){
        return x+y;
    }

    //Divide
    public double mDivide(int x, int y){
        return x/y;
    }
}
```



ECLIPSE

Installing Java and Eclipse on Ubuntu Linux: 2013 Updates

Do you have a JDK installed? You likely want to put $JDK_HOME/bin on your PATH, not the /bin of a JRE, as jar comes with JDK, not JRE.

Do this:

1. Delete all installations of Java.
2. Install the Java SDK (self-extracting) into **/opt/jdk1.6.0_16** (for example)
3. Create a symbolic link: ln -s /opt/jdk1.6.0_16 /opt/jdk

4. Edit $HOME/.bashrc:

```
JAVA_HOME=/opt/jdk
PATH=$PATH:$HOME/bin:$JAVA_HOME/bin
```

5. Logout and log back in.

This offers many advantages:

- You can install multiple versions of the SDK and need only switch a symbolic link.
- You know where all the files are located.
- You know exactly which version of Java is being used.
- No other versions are installed, so there cannot be any conflicts.

I have done this for years and have never had any problems with Java on Linux, except for packages that do not detect that Java is installed and attempt to install the OpenJDK.

Also, stay away from the OpenJDK as its fonts are terrible to behold.

Environment Preparation

- JavaSE 6r29 JDK(and JRE): java.oracle.com
- Set Path & Test: “C:\Program Files\Java\jre6\bin”
  - java -version
  - javac -version
- IDE
  - Eclipse: www.eclipse.org/downloads
    - Eclipse IDE for Java Developers/Jave EE Developers
  -

JARs(include in Eclipse project when needed):

- TestND: http://beust.com/eclipse
- Java Excel API: http://jexcelapi.sourceforge.net/
  - http://jexcelapi.sourceforge.net/resources/javadocs/2_3_8/docs/jxl/read/biff/BiffException.html
- JUnit: http://www.junit.org/
- Apache ANT (HTML reporting)
- Subversion http://subclipse.tigris.org/update_1.8.x
- Also check: http://download.eclipse.org/releases/juno/

Help and Tutorials

Java/Selenium Reading from Excel sheets:

- http://testerinyou.blogspot.com/2010/10/how-to-do-data-driven-testing-using.html
- http://functionaltestautomation.blogspot.com/2009/10/dataprovider-data-driven-testing-with.html
- http://www.youtube.com/watch?v=ty3q2wQdPmU&feature=BFp&list=WL18EEBB0491EF4A05

Shortcuts:

CTRL-SPACE	Code writing
CTRL + /	Add/remove comments
CTRL + SHFT + O	Organize imports (Add/remove imports)
ALT + UP/DOWN	Move a line of code up or down
CTRL + D	Delete Line
ALT+SHFT+J	Element Comment

Eclipse Start.bat: JDK7

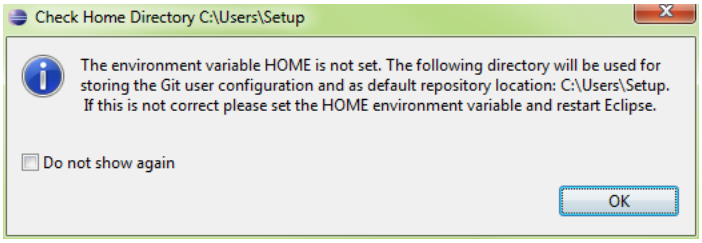
This uses the JDK7 binaries:

```
set DEV_HOME=D:\DEVL\Java
set JAVA_HOME=%DEV_HOME%\java32\jdk7
set PATH=%JAVA_HOME%\bin;%PATH%

start %DEV_HOME%\eclipse32\eclipse.exe -vm %JAVA_HOME%\bin\javaw.exe -showlocation -
vmargs -server -Xms512m -Xmx1024m -XX:MaxPermSize=128m
```

Eclipse.ini Configuration: JDK7

```
-startup
plugins/org.eclipse.equinox.launcher_1.3.0.v20120522-1813.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_1.1.200.v20120522-1813
-product
org.eclipse.epp.package.java.product
--launcher.defaultAction
openFile
--launcher.XXMaxPermSize
256M
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
256m
--launcher.defaultAction
openFile
-vm
D:/DEVL/Java/java32/java32/jdk7/bin/javaw.exe
-vmargs
-Dosgi.requiredJavaVersion=1.5
-Dhelp.lucene.tokenizer=standard
-Xms40m
-Xmx512m
```

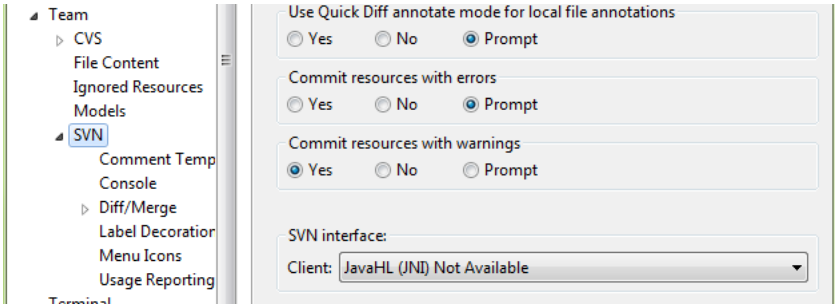
HOME Environment Variable



Installing the Subversion Client

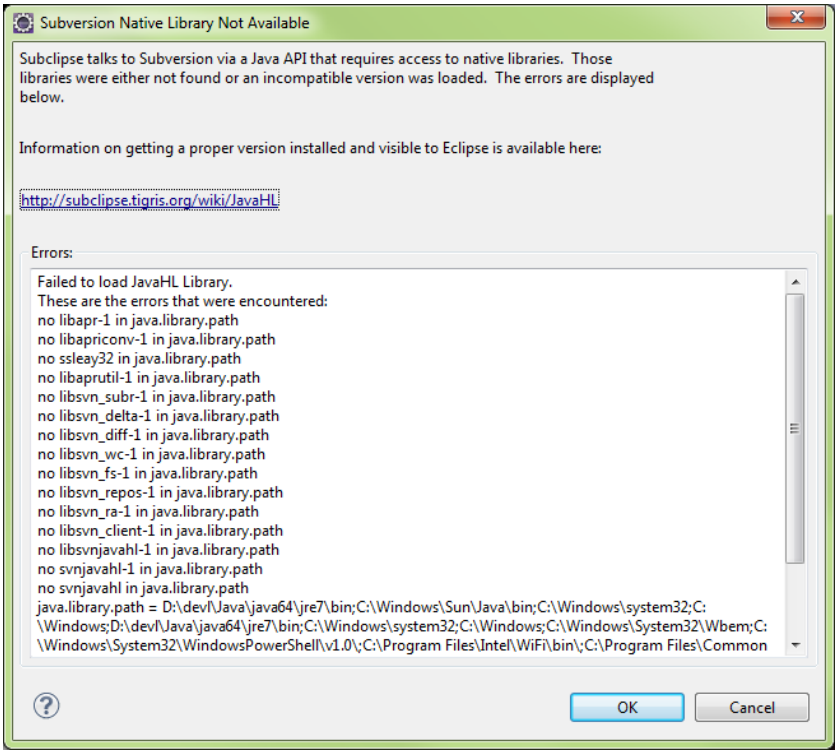
- Go to Help\Eclipse Marketplace and search by 'Subversion'
- Select Subclipse and Install...
- http://subclipse.tigris.org/update_1.8.x

- Go to Preferences\Team\SVN
- The SVN interface should be JavaHL(JNI)
- Here it is Not Available



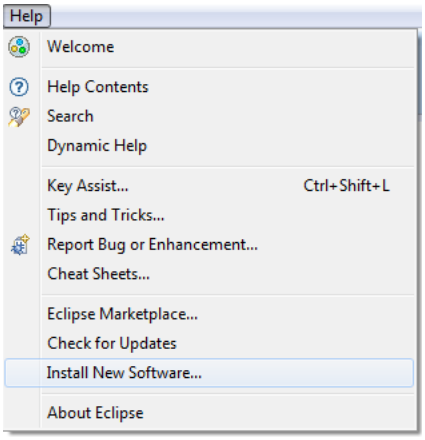
- If the Subclipse client plugins are missing, the following prompt will appear

Subclipse Library Not Available



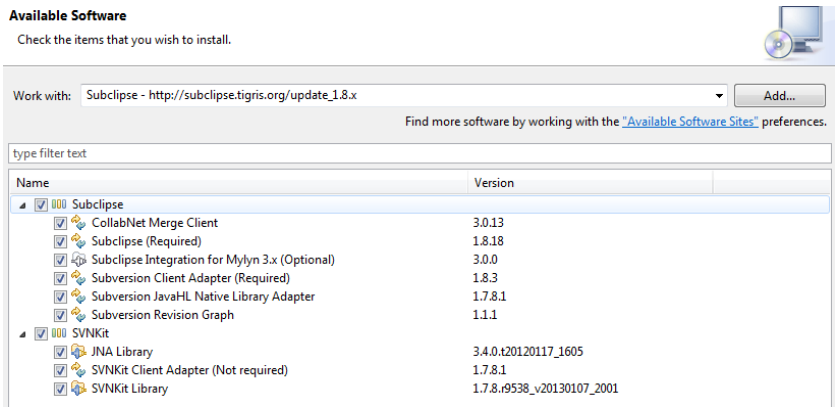
Installing Subclipse from Help\New Software

- Go to Help\Install New Software...

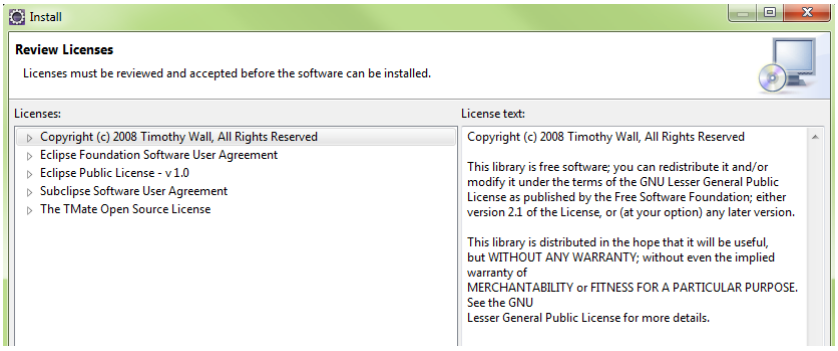


Locate "Subclipse – http://..."

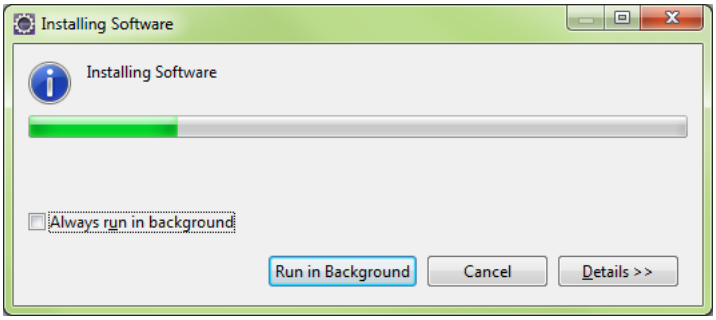
Select Subclipse option, Click Next



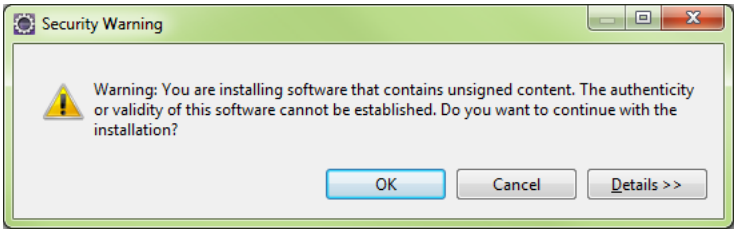
Acknowledge EULAs



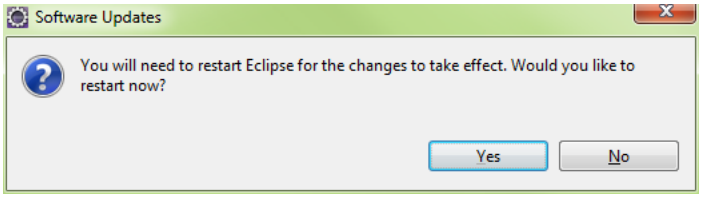
Eclipse will download selected packages



Security Warning

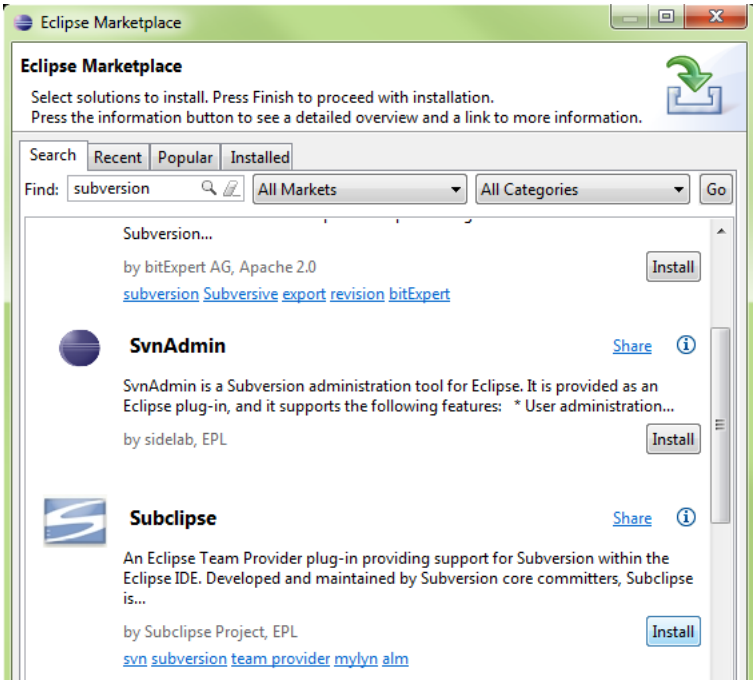


Eclipse will restart

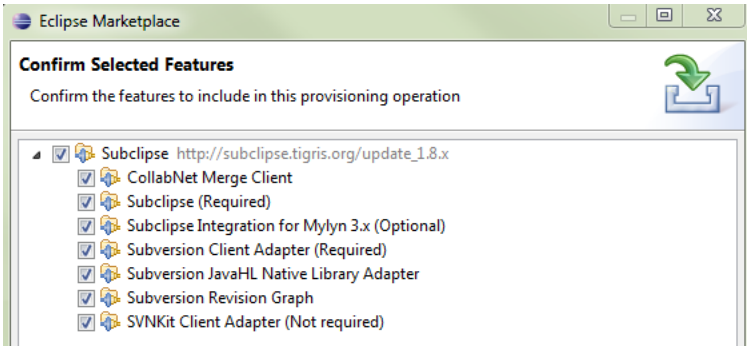


Installing Subclipse From Help\Eclipse Market Place…

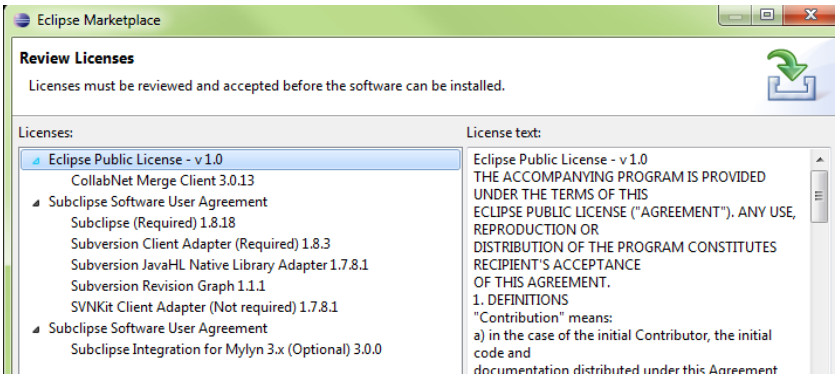
Eclipse Marketplace: Subclipse selection



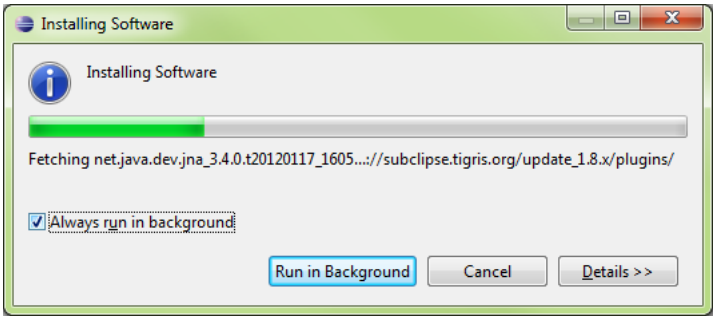
Select all packages



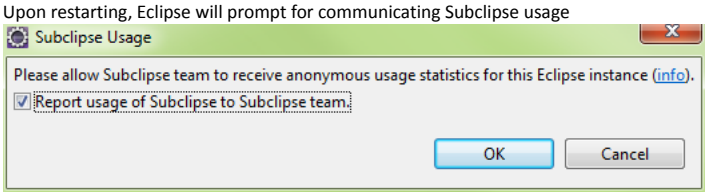
Acknowledge the EULAs



Eclipse will download the packages



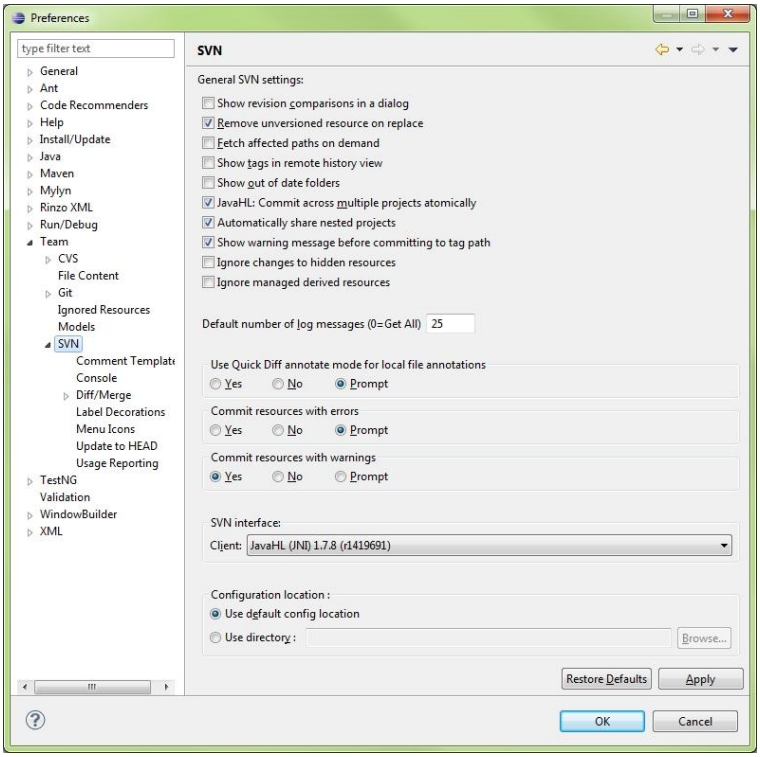
Upon restarting, Eclipse will prompt for communicating Subclipse usage



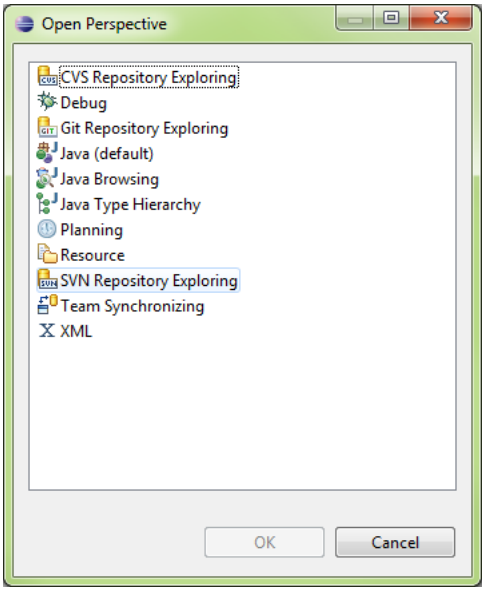
Enabling the SVN Connector and Plugin

Eclipse\Preferences\Team\SVN

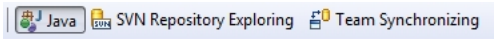
Select SVN interface: JavaHL and press OK



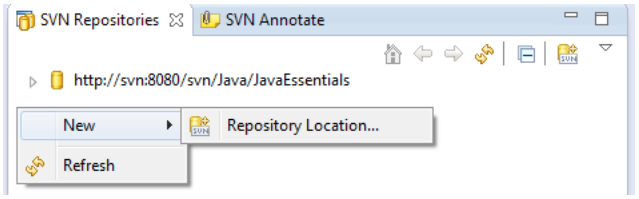
Open the SVN Repository View



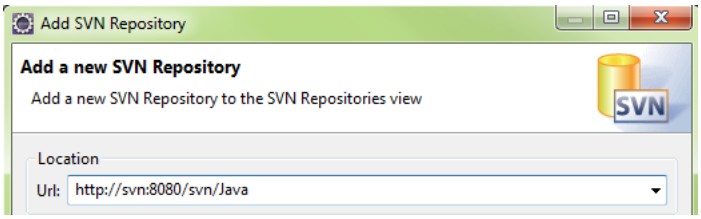
The SVN repository Exporting Panel will appear



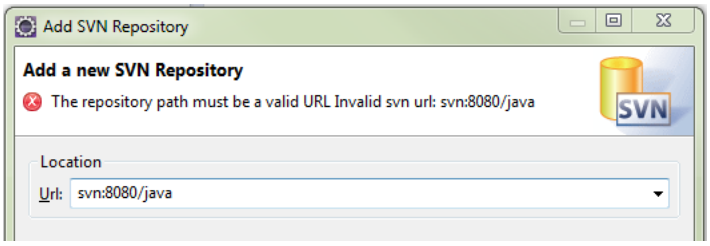
Right Click in the SVN repositories panel, select New..



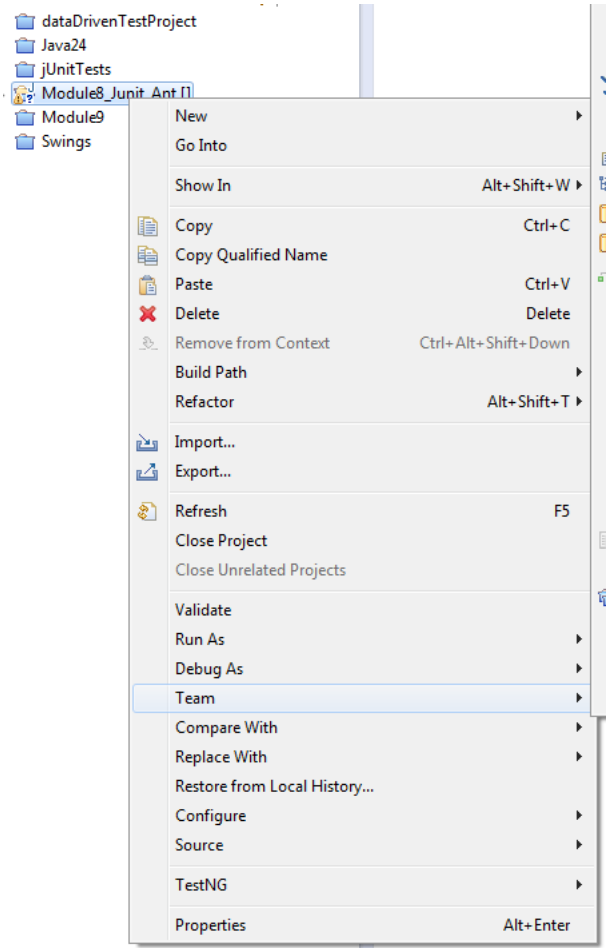
Enter the URL of the SVN Server and Repository



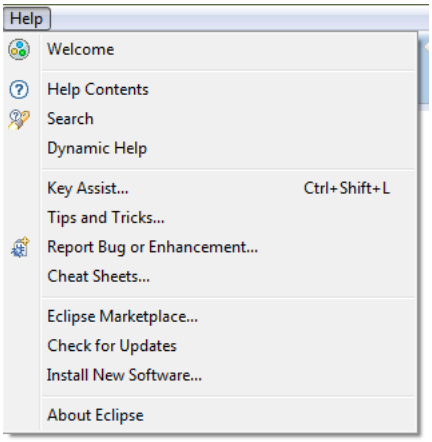
If Incorrect, Eclipse will indicate an Error



Accessing SVN from the Project Explorer

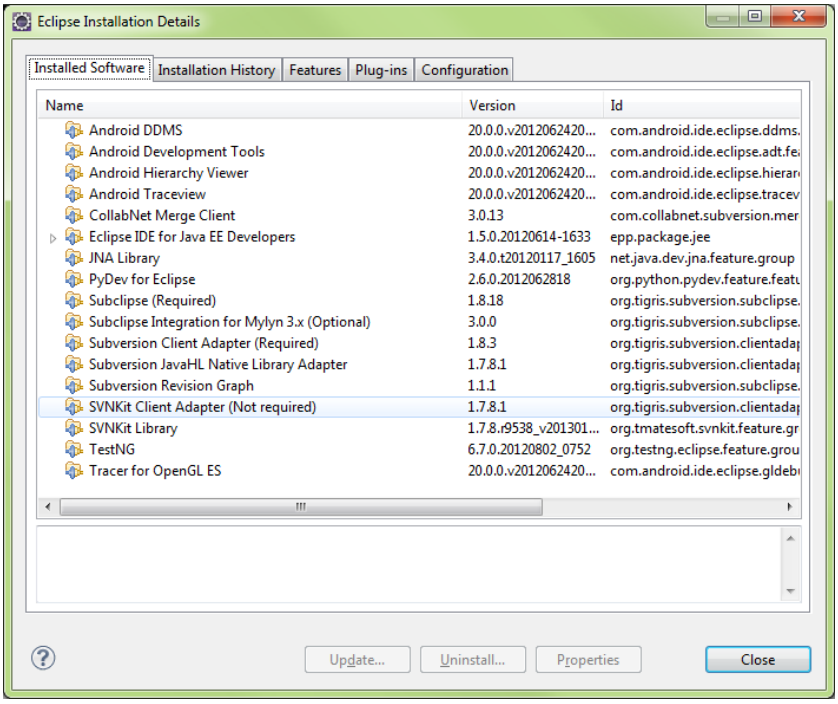


Eclipse About

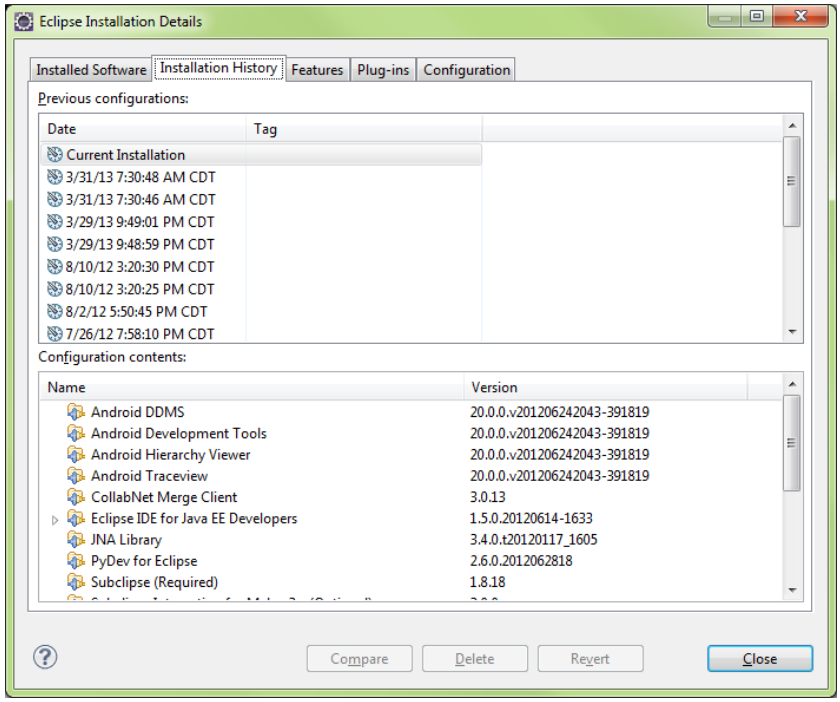


About Main Form

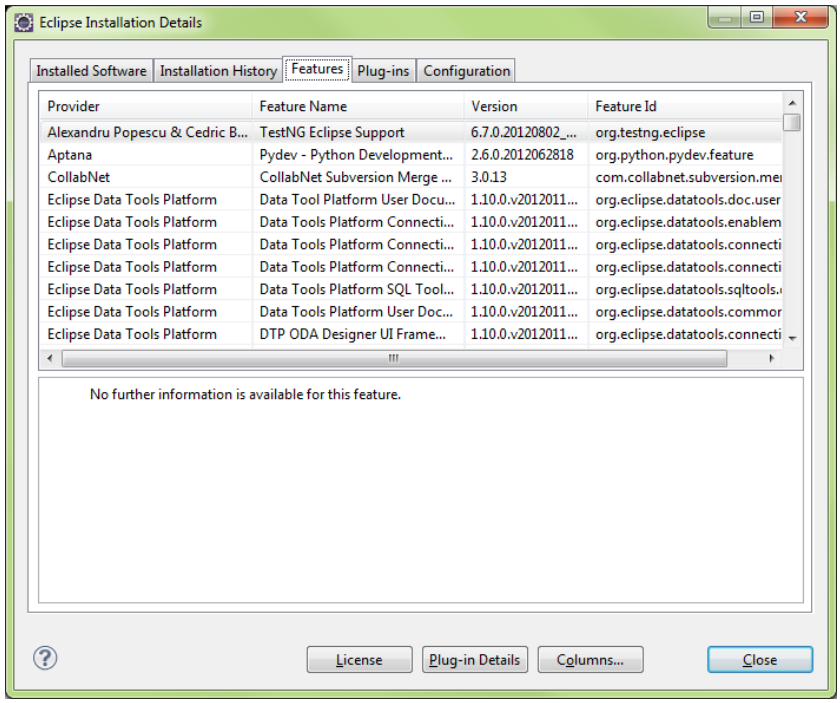
Installation Details



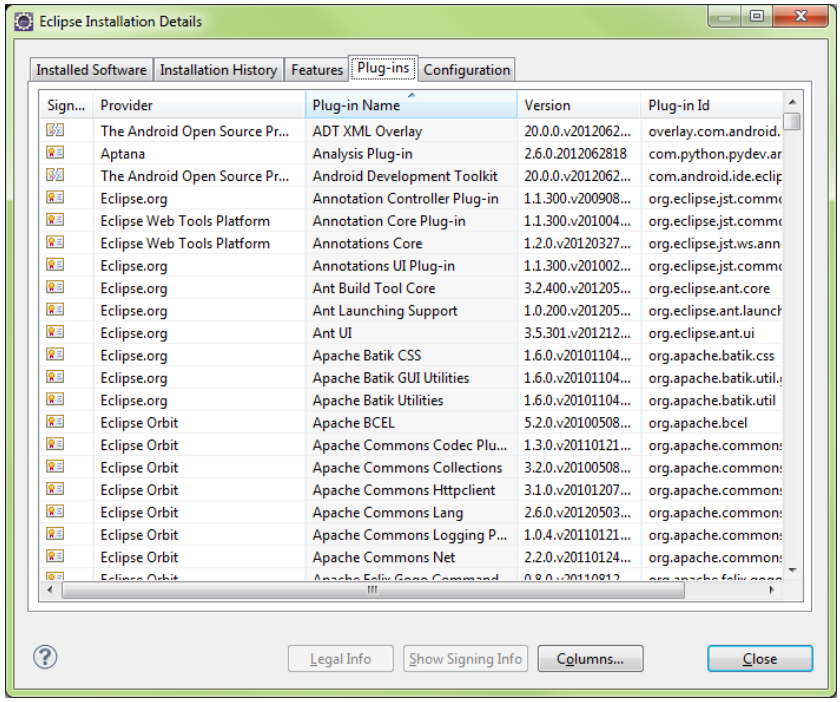
Installation History



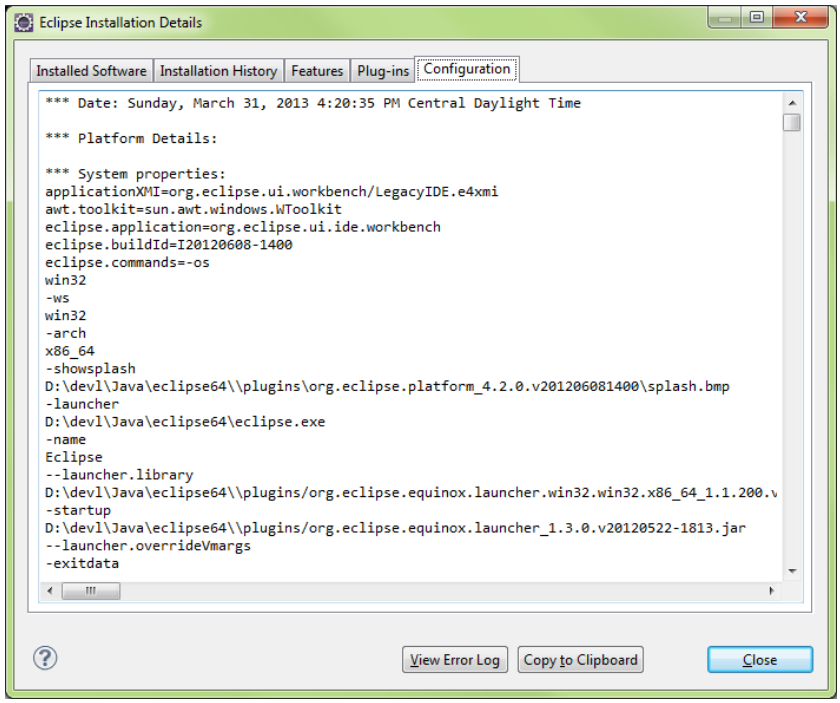
Features



Plugins

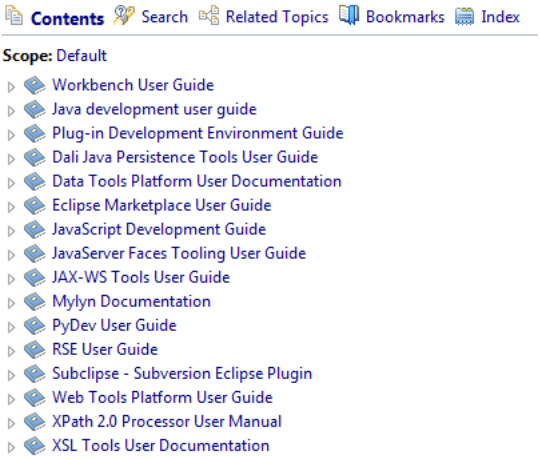


Eclipse Installation Details

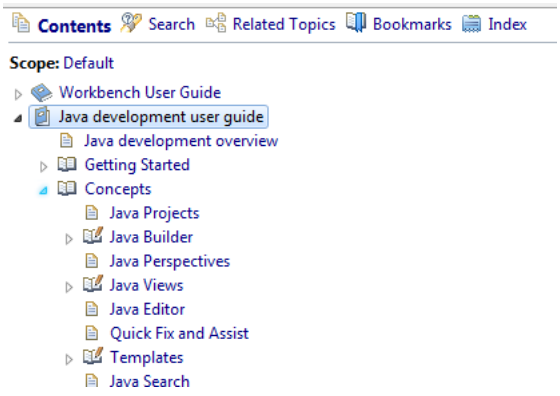


Eclipse Help

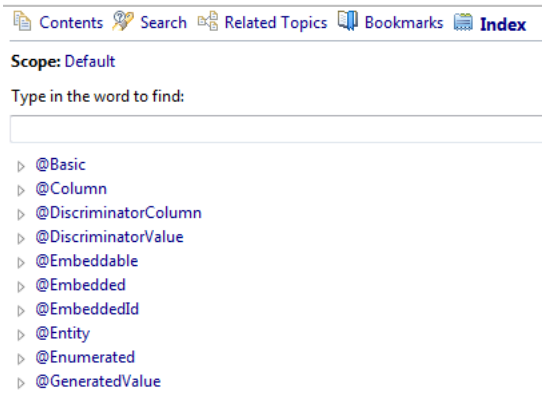
Main Contents View



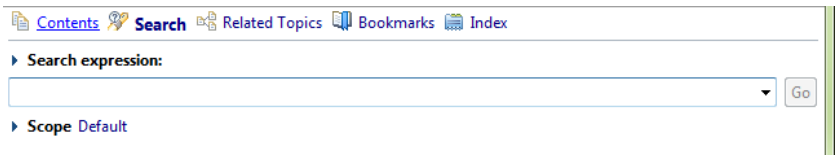
Help Contents Expanded



Help Index



Help Search



Compiling and Running

Java package statement implies the directory structure where it exists within the project.
Should be unique

- package **com**.lynda.javatraining;
- \src**com**\lynda\javatraining**HelloWorld**.java

When compiling, use javac in the project root:
- C:\JavaProjects\HelloWorld>javac com\lynda\javatraining\HelloWorld.java
  - HelloWorld.class
  - HelloWorld.java

When running, use package reference and filename **without** ".java" extension
- C:\JavaProjects\HellowWorld>java **com**.lynda.javatraining.**HelloWorld**

public class **HelloWorld** {

public static void main(String[] args) {
- **Static: allows class to be called directly?**

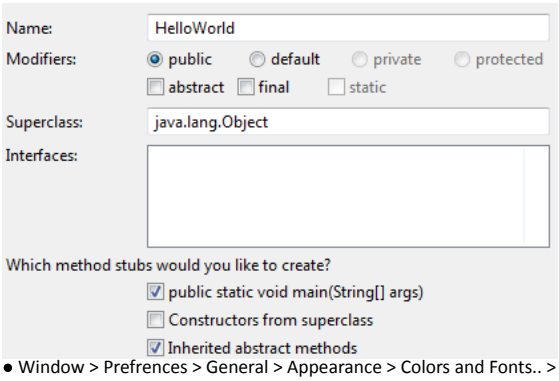
public class **HelloWorld** {

public static void main(String[] args) {
- **Static: allows class to be called directly?**

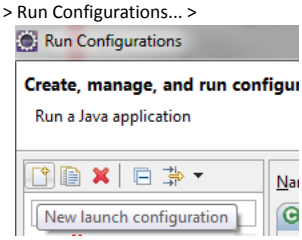
Eclipse Development

Create Workspace

- Create WORKSPACE folder to contain PROJECTS
- New, Project....
- New, Class....(.java file is the class)



- Window > Prefrences > General > Appearance > Colors and Fonts.. > Text Font
- Run > Run Configurations... >

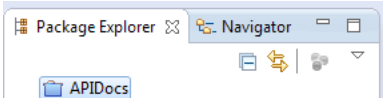


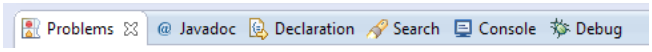
Import Projects

- File > Import > General > Existing Project into Workspace
  - Exploded Project (Root Dir)
  - Zipped Project (Archive File)
  - Root Dir, OK
  - Eclipse will autodetect projects
  - Copy projects into workspace
- Close projects that aren't in use (Right-Click, Close Project)
  - Projects remain in workspace, just not open

IDE Layout

HutuBBB
Views/Panels





Perspectives

- Arrangement of Views
- 

- Custom Perspectives: **Window > Save Perspective As....**

Command Line

- Dir to PROJECT\SRC
- `javac Main.java`
  - `dir = Main.class`
  - `java Main`
- `javac Main.java -d ..\bin`
  - `\src\Main.class`
- `javac Main.java -verbose`

Note: Access Denied Errors

If **Access Denied** error occurs, set folder permissions to Everyone and give **Full Control** access.

```
D:\Eclipse\Java\CommandLine\src>javac Main.java -d D:\Eclipse\Java\CommandLine\bin
Main.java:2: error while writing Main: D:\Eclipse\Java\CommandLine\bin\Main.class
(Access is denied

public class Main {
             ^
```

ClassPath

Similar to the classic dynamic loading behavior, when executing Java programs, the Java Virtual Machine finds and loads classes lazily (it loads the bytecode of a class only when this class is first used). The classpath tells Java where to look in the filesystem for files defining these classes.
The virtual machine searches for and loads classes in this order:

1 bootstrap classes: the classes that are fundamental to the Java Platform (comprising the public classes of the Java Class Library, and the private classes that are necessary for this library to be functional).
2 extension classes: packages that are in the *extension* directory of the JRE or JDK, jre/lib/ext/
3 user-defined packages and libraries

By default only the packages of the JDK standard API and extension packages are accessible without needing to set where to find them. The path for all user-defined packages and libraries must be set in the command-line (or in the Manifest associated with the Jar file containing the classes).

Setting the path through an environment variable

The environment variable named CLASSPATH may be alternatively used to set the classpath. For the above example, we could also use on Windows:
Sometimes you have to check the JAVA_HOME also, if it is pointing towards the right JDK version
set CLASSPATH=D:\myprogram
java org.mypackage.HelloWorld

Diagnose

Application programmers may want to find out/debug the current settings under which the application is running:
System.getProperty("java.class.path")

XXX