DrJava

Appendix A

Barb Ericson
Georgia Institute of Technology
May 2006

Learning Goals

- Understand at practical level
  - Where to get DrJava
  - How to start DrJava
  - Dr Java features
  - How to add items to the classpath for DrJava
  - How to use the panes
  - How to compile and run the main method
  - How to generate Javadoc html pages
  - How to use the debugger
  - How to create and run JUnit tests

Where to get DrJava

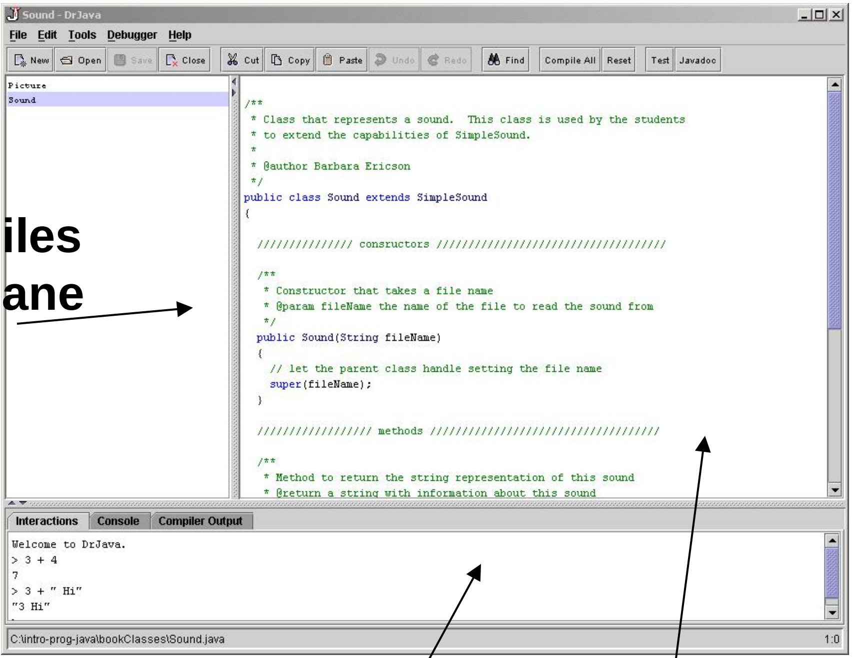
- DrJava is a free development environment for Java aimed at students
  - From Rice University
- It can be downloaded from
  - http://www.drjava.org/
- It requires Java
  - We recommend using 1.5 (5.0)
  - Download Java first from java.sun.com

How to Start DrJava

- Click on the DrJava icon on your desktop
- Wait while DrJava loads
  - It sill display the splash screen while loading
- Once you see the full environment
  - you may begin

DrJava Features

- Works with multiple files
  - – Files pane
- Color coded editor
  - – Definitions pane
- Interpretation of Java code
  - – Interactions pane
- Integrated debugger



```
/**
 * Class that represents a sound.  This class is used by the students
 * to extend the capabilities of SimpleSound.
 *
 * @author Barbara Ericson
 */
public class Sound extends SimpleSound
{

  /////////////// consructors ////////////////////////////////////

  /**
   * Constructor that takes a file name
   * @param fileName the name of the file to read the sound from
   */
  public Sound(String fileName)
  {
    // let the parent class handle setting the file name
    super(fileName);
  }

  ////////////////// methods ///////////////////////////////////

  /**
   * Method to return the string representation of this sound
   * @return a string with information about this sound
```

```
Welcome to DrJava.
> 3 + 4
7
> 3 + " Hi"
"3 Hi"
```

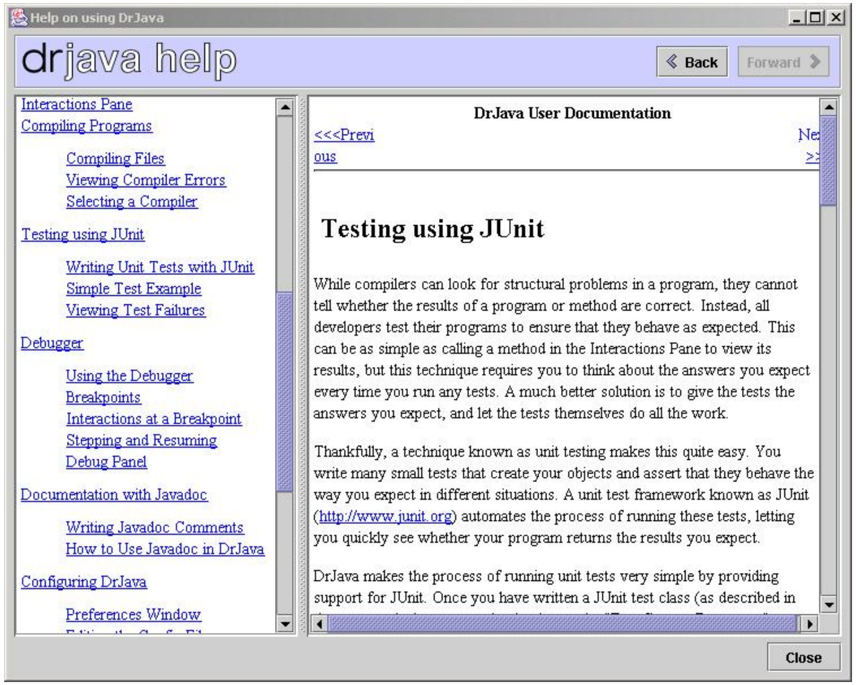
Files pane

Interactions pane

Definitions pane

Help Window

- Click on Help in the menu and then again on Help to bring up the help window
  - Or use F1
- The table of contents is on the left
  - Click on a topic to see the help on the right



Interactions Pane
Compiling Programs

Compiling Files
Viewing Compiler Errors
Selecting a Compiler

Testing using JUnit

Writing Unit Tests with JUnit
Simple Test Example
Viewing Test Failures

Debugger

Using the Debugger
Breakpoints
Interactions at a Breakpoint
Stepping and Resuming
Debug Panel

Documentation with Javadoc

Writing Javadoc Comments
How to Use Javadoc in DrJava

Configuring DrJava

Preferences Window

DrJava User Documentation

<<<Previ
ous

Ne
>>

Testing using JUnit

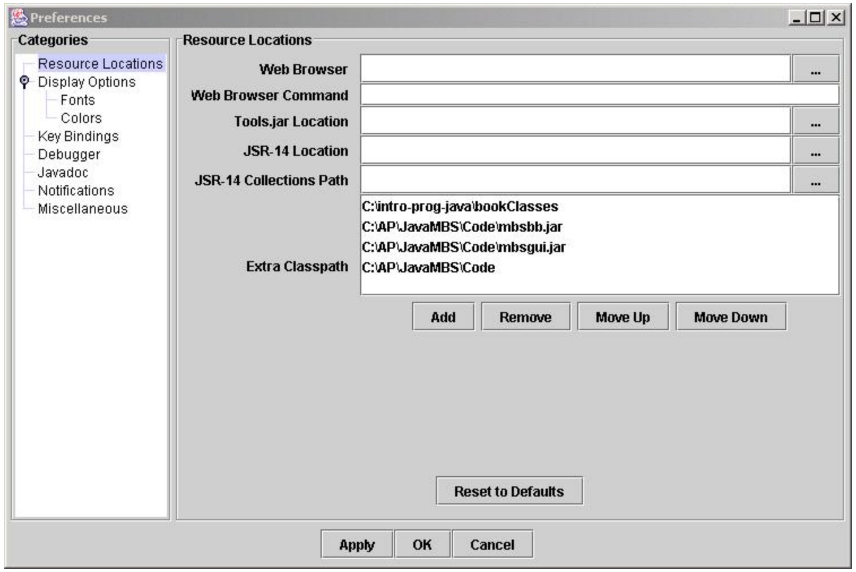
While compilers can look for structural problems in a program, they cannot tell whether the results of a program or method are correct. Instead, all developers test their programs to ensure that they behave as expected. This can be as simple as calling a method in the Interactions Pane to view its results, but this technique requires you to think about the answers you expect every time you run any tests. A much better solution is to give the tests the answers you expect, and let the tests themselves do all the work.

Thankfully, a technique known as unit testing makes this quite easy. You write many small tests that create your objects and assert that they behave the way you expect in different situations. A unit test framework known as JUnit (http://www.junit.org) automates the process of running these tests, letting you quickly see whether your program returns the results you expect.

DrJava makes the process of running unit tests very simple by providing support for JUnit. Once you have written a JUnit test class (as described in

How to add to the classpath

- Bring up the preferences window
  - Click on Edit and then Preferences
  - Click the add button
    - And select all jar files that you want to add to the classpath
    - Also add directories that have classes that you wish to add to the classpath
    - When you are done click on “Ok”



Extra Classpath

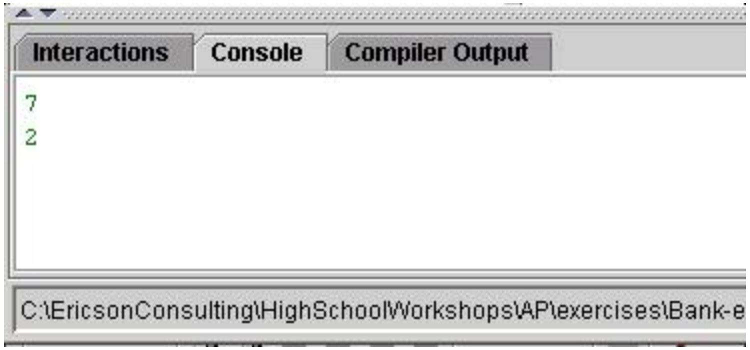
```
C:\intro-prog-java\bookClasses
C:\AP\JavaMBS\Code\mbsbb.jar
C:\AP\JavaMBS\Code\mbsgui.jar
C:\AP\JavaMBS\Code
```

How to use the interactions pane

- If you don’t end a statement with a semicolon ‘;’ it will be interpreted and the result printed
- If you end a statement with a semicolon it will be interpreted but the result won’t be printed
  - Use System.out.println(expression) to print
- To enter multiple line statements use Shift + Enter

How to use the console pane

- If you click on the console tab
  - You will see the console pane
- It shows just the items that you have printed to the console
  - Using System.out.println or
  - System.out.print

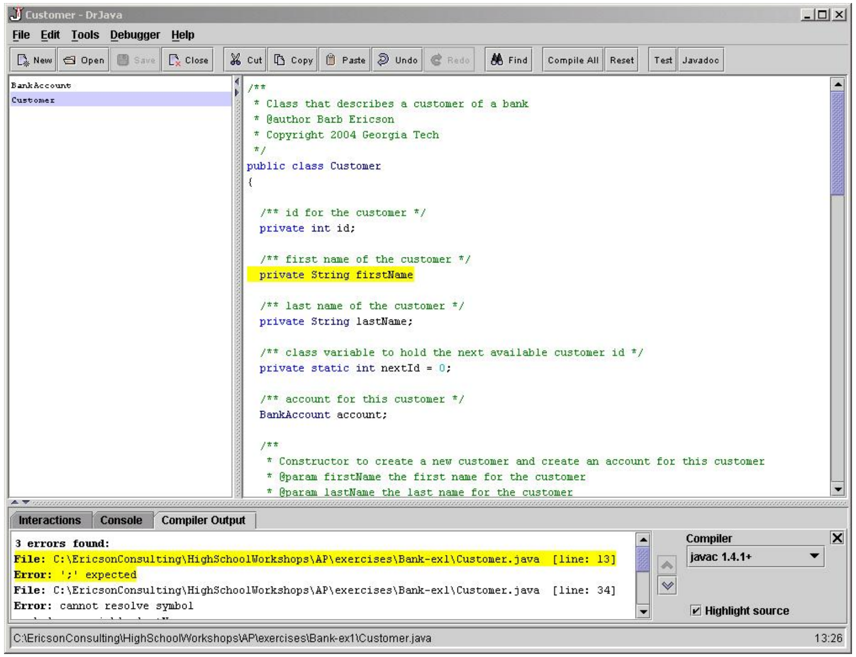


```
7
2
```

C:\EricsonConsulting\HighSchoolWorkshops\AP\exercises\Bank-e

Compiler output pane

- Compiler errors are shown in the compiler output pane
  - The first error will be highlighted and the line of code that caused the problem will be highlighted
  - You can also click on an error to go to the line that contains the error


Customer - DrJava
File Edit Tools Debugger Help
New Open Save Close Cut Copy Paste Undo Redo Find Compile All Reset Test Javadoc
BankAccount
Customer
Interactions Console Compiler Output
Compiler
javac 1.4.1+
☑ Highlight source

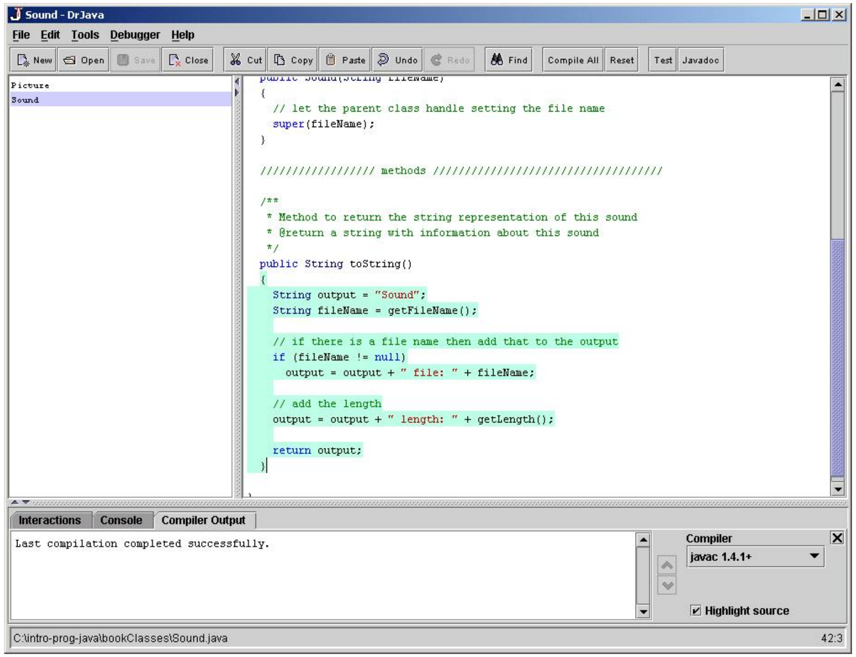
```
/**
 * Class that describes a customer of a bank
 * @author Barb Ericson
 * Copyright 2004 Georgia Tech
 */
public class Customer
{

  /** id for the customer */
  private int id;

  /** first name of the customer */
  private String firstName

  /** last name of the customer */
  private String lastName;

  /** class variable to hold the next available customer id */
  private static int nextId = 0;

  /** account for this customer */
  BankAccount account;

  /**
   * Constructor to create a new customer and create an account for this customer
   * @param firstName the first name for the customer
   * @param lastName the last name for the customer
```

```
3 errors found:
File: C:\EricsonConsulting\HighSchoolWorkshops\AP\exercises\Bank-ex1\Customer.java  [line: 13]
Error: ';' expected
File: C:\EricsonConsulting\HighSchoolWorkshops\AP\exercises\Bank-ex1\Customer.java  [line: 34]
Error: cannot resolve symbol
```

C:\EricsonConsulting\HighSchoolWorkshops\AP\exercises\Bank-ex1\Customer.java 13:26

How to use the definitions pane

- Click in the definitions pane to add code to a file
  - It will automatically indent for you when you hit enter
    - Or use tab
  - It will highlight all code in a block when you click to the right of a closing parenthesis
  - You can indent selected lines using the Edit menu
  - You can comment out and uncomment lines in the Edit menu



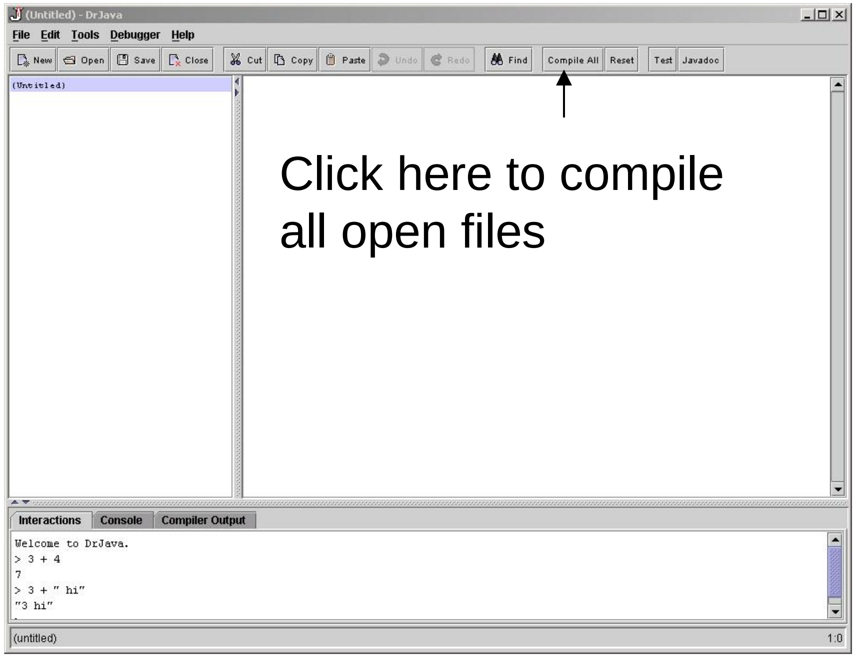
```
public Sound(String fileName)
{
  // let the parent class handle setting the file name
  super(fileName);
}

////////////////// methods ////////////////////////////////////

/**
 * Method to return the string representation of this sound
 * @return a string with information about this sound
 */
public String toString()
{
  String output = "Sound";
  String fileName = getFileName();

  // if there is a file name then add that to the output
  if (fileName != null)
    output = output + " file: " + fileName;

  // add the length
  output = output + " length: " + getLength();

  return output;
}
```

```
Last compilation completed successfully.
```

C:\intro-prog-java\bookClasses\Sound.java

How to compile programs

- Click on Compile All to compile all files in the files pane
  - Or use Tools->Compile All Documents
- You can compile just the shown file by using
  - Tools->Compile Current Document

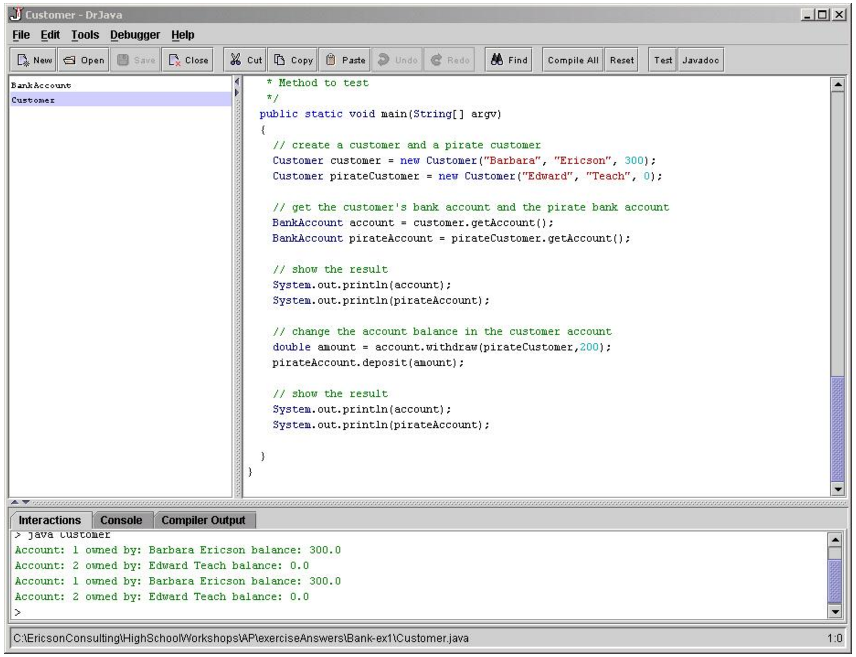


```
Welcome to DrJava.
> 3 + 4
7
> 3 + " hi"
"3 hi"
```

Click here to compile all open files

How to execute programs

- Make sure that the file that you want to run the main method from is selected in the files pane
- Click on Tools->Run Document’s Main Method
- Output will be shown in the interactions and console panes



```
   * Method to test
   */
  public static void main(String[] argv)
  {
    // create a customer and a pirate customer
    Customer customer = new Customer("Barbara", "Ericson", 300);
    Customer pirateCustomer = new Customer("Edward", "Teach", 0);

    // get the customer's bank account and the pirate bank account
    BankAccount account = customer.getAccount();
    BankAccount pirateAccount = pirateCustomer.getAccount();

    // show the result
    System.out.println(account);
    System.out.println(pirateAccount);

    // change the account balance in the customer account
    double amount = account.withdraw(pirateCustomer,200);
    pirateAccount.deposit(amount);

    // show the result
    System.out.println(account);
    System.out.println(pirateAccount);

  }
}
```

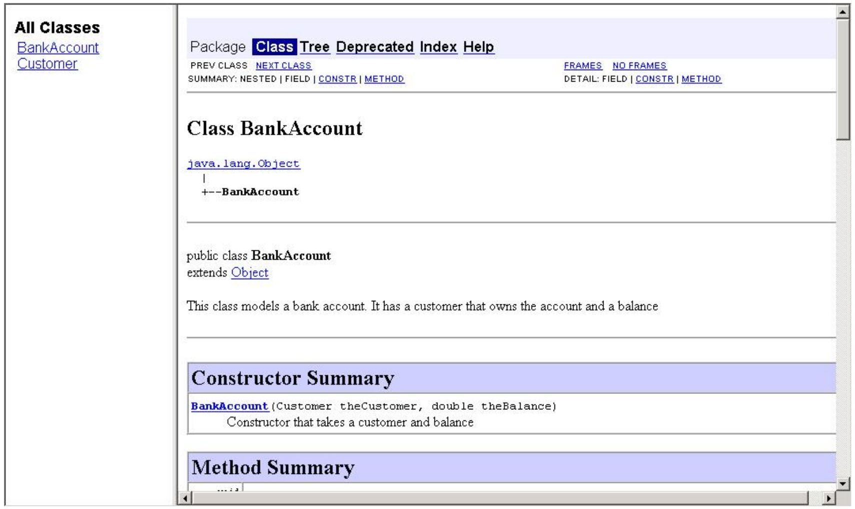
```
> java Customer
Account: 1 owned by: Barbara Ericson balance: 300.0
Account: 2 owned by: Edward Teach balance: 0.0
Account: 1 owned by: Barbara Ericson balance: 300.0
Account: 2 owned by: Edward Teach balance: 0.0
>
```

C:\EricsonConsulting\HighSchoolWorkshops\AP\exerciseAnswers\Bank-ex1\Customer.java 1:0

How to generate Javadoc documents

- Click on the Javadoc button
  - It will generate the Javadoc documentation for all files in the folder and all subfolders for all open files
  - From javadoc comments

    /** comment */


All Classes
BankAccount
Customer
Package Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS
FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD

Class BankAccount

```
java.lang.Object
  |
  +--BankAccount
```

public class **BankAccount**
extends Object

This class models a bank account. It has a customer that owns the account and a balance

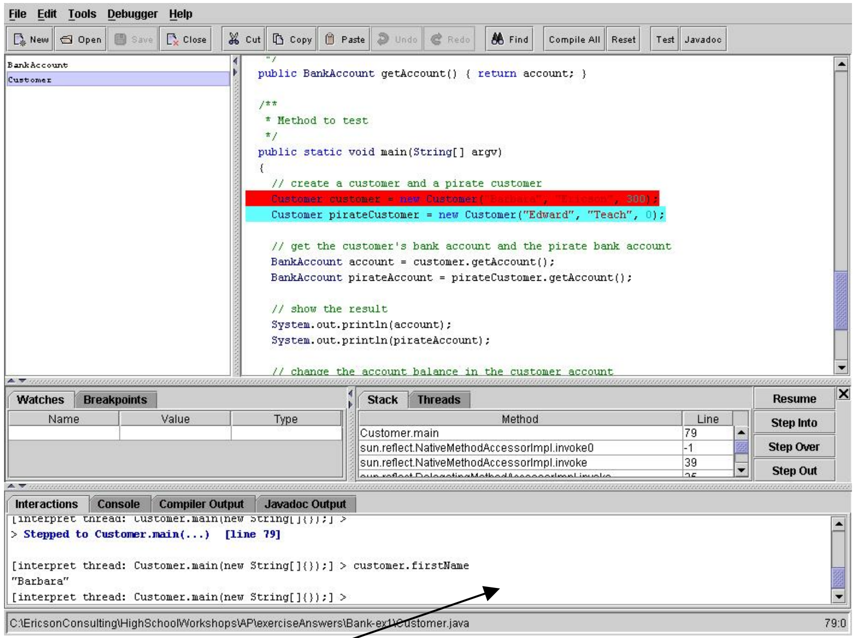
Constructor Summary

`BankAccount(Customer theCustomer, double theBalance)`
Constructor that takes a customer and balance

Method Summary

How to use the debugger

- Click Debugger->Debug Mode
- Select a line of code to stop at "breakpoint"
  - Right click and Toggle Breakpoint
  - You can have several breakpoints
- Run the main method
- Use the interactions pane to explore the variables



```
   */
  public BankAccount getAccount() { return account; }

  /**
   * Method to test
   */
  public static void main(String[] argv)
  {
    // create a customer and a pirate customer
    Customer customer = new Customer("Barbara", "Ericson", 300);
    Customer pirateCustomer = new Customer("Edward", "Teach", 0);

    // get the customer's bank account and the pirate bank account
    BankAccount account = customer.getAccount();
    BankAccount pirateAccount = pirateCustomer.getAccount();

    // show the result
    System.out.println(account);
    System.out.println(pirateAccount);

    // change the account balance in the customer account
```

Name	Value	Type

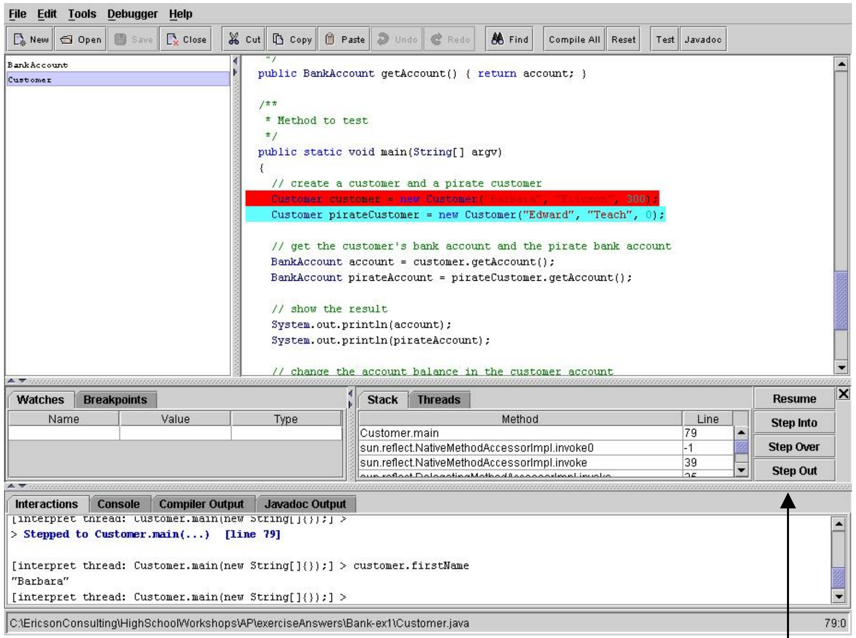
Method	Line
Customer.main	79
sun.reflect.NativeMethodAccessorImpl.invoke0	-1
sun.reflect.NativeMethodAccessorImpl.invoke	39
sun.reflect.DelegatingMethodAccessorImpl.invoke	25

```
[interpret thread: Customer.main(new String[]{});] >
> Stepped to Customer.main(...)  [line 79]

[interpret thread: Customer.main(new String[]{});] > customer.firstName
"Barbara"
[interpret thread: Customer.main(new String[]{});] >
```

C:\EricsonConsulting\HighSchoolWorkshops\AP\exerciseAnswers\Bank-ex1\Customer.java 79:0

Debug Stepping and Resume

- Use Step Into
  - To see each piece of the current line being executed.
  - Goes into methods
- Use Step Over
  - To execute the current line without seeing the execution and stop before the next line is executed
- Use Step Out
  - To execute the rest of the current method and stop before the next line in the calling method
- Use Resume
  - To continue execution still the next breakpoint or the end of the program


File Edit Tools Debugger Help
New Open Save Close Cut Copy Paste Undo Redo Find Compile All Reset Test Javadoc
BankAccount
Customer
Watches Breakpoints
Stack Threads
Resume
Step Into
Step Over
Step Out
Interactions Console Compiler Output Javadoc Output

```
  */
public BankAccount getAccount() { return account; }

/**
 * Method to test
 */
public static void main(String[] argv)
{
  // create a customer and a pirate customer
  Customer customer = new Customer("Barbara", "Ericson", 300);
  Customer pirateCustomer = new Customer("Edward", "Teach", 0);

  // get the customer's bank account and the pirate bank account
  BankAccount account = customer.getAccount();
  BankAccount pirateAccount = pirateCustomer.getAccount();

  // show the result
  System.out.println(account);
  System.out.println(pirateAccount);

  // change the account balance in the customer account
```

Name	Value	Type

Method	Line
Customer.main	79
sun.reflect.NativeMethodAccessorImpl.invoke0	-1
sun.reflect.NativeMethodAccessorImpl.invoke	39
sun.reflect.DelegatingMethodAccessorImpl.invoke	25

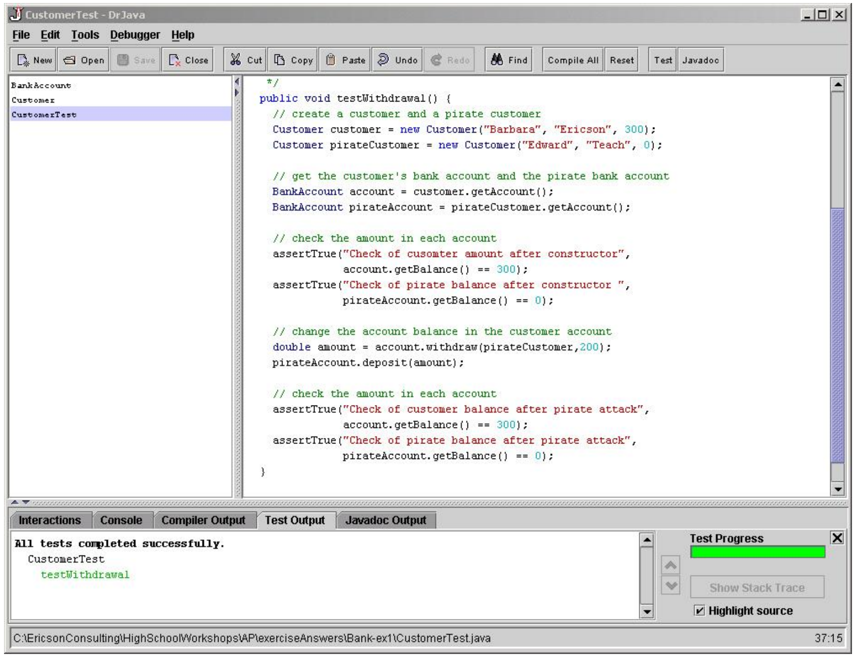
```
[interpret thread: Customer.main(new String[]{});] >
> Stepped to Customer.main(...)  [line 79]

[interpret thread: Customer.main(new String[]{});] > customer.firstName
"Barbara"
[interpret thread: Customer.main(new String[]{});] >
```

C:\EricsonConsulting\HighSchoolWorkshops\AP\exerciseAnswers\Bank-ex1\Customer.java 79:0

Stepping Options

How to create JUnit tests

- Click on “Edit” then "New JUnit Test Case"
- Create one or more test*Something* methods
  - assertTrue(String, boolean)
  - assertEquals(String,int ,int)
  - fail(String)



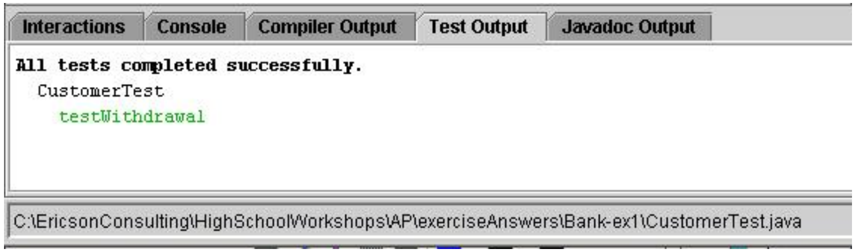
```
  */
public void testWithdrawal() {
  // create a customer and a pirate customer
  Customer customer = new Customer("Barbara", "Ericson", 300);
  Customer pirateCustomer = new Customer("Edward", "Teach", 0);

  // get the customer's bank account and the pirate bank account
  BankAccount account = customer.getAccount();
  BankAccount pirateAccount = pirateCustomer.getAccount();

  // check the amount in each account
  assertTrue("Check of cusomter amount after constructor",
             account.getBalance() == 300);
  assertTrue("Check of pirate balance after constructor ",
             pirateAccount.getBalance() == 0);

  // change the account balance in the customer account
  double amount = account.withdraw(pirateCustomer,200);
  pirateAccount.deposit(amount);

  // check the amount in each account
  assertTrue("Check of customer balance after pirate attack",
             account.getBalance() == 300);
  assertTrue("Check of pirate balance after pirate attack",
             pirateAccount.getBalance() == 0);
}
```

```
All tests completed successfully.
  CustomerTest
    testWithdrawal
```

C:\EricsonConsulting\HighSchoolWorkshops\AP\exerciseAnswers\Bank-ex1\CustomerTest.java 37:15

How to run JUnit tests

- Click the Test button
  - Results are shown in the Test Output pane
- All failures will be reported
- JUnit tests are very useful for initial testing
  - And regression testing



```
All tests completed successfully.
  CustomerTest
    testWithdrawal
```

C:\EricsonConsulting\HighSchoolWorkshops\AP\exerciseAnswers\Bank-ex1\CustomerTest.java

Summary

- DrJava is a free, lightweight, development environment
    - – Designed for students
- You can use it to create, compile, debug and execute Java programs
- You can use it to generate Javadoc documentation
- You can use it to create and run JUnit tests