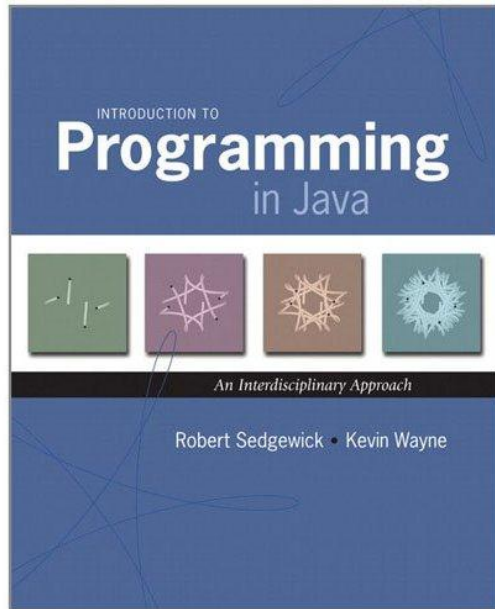


Introduction to Computer Science and Programming in Java

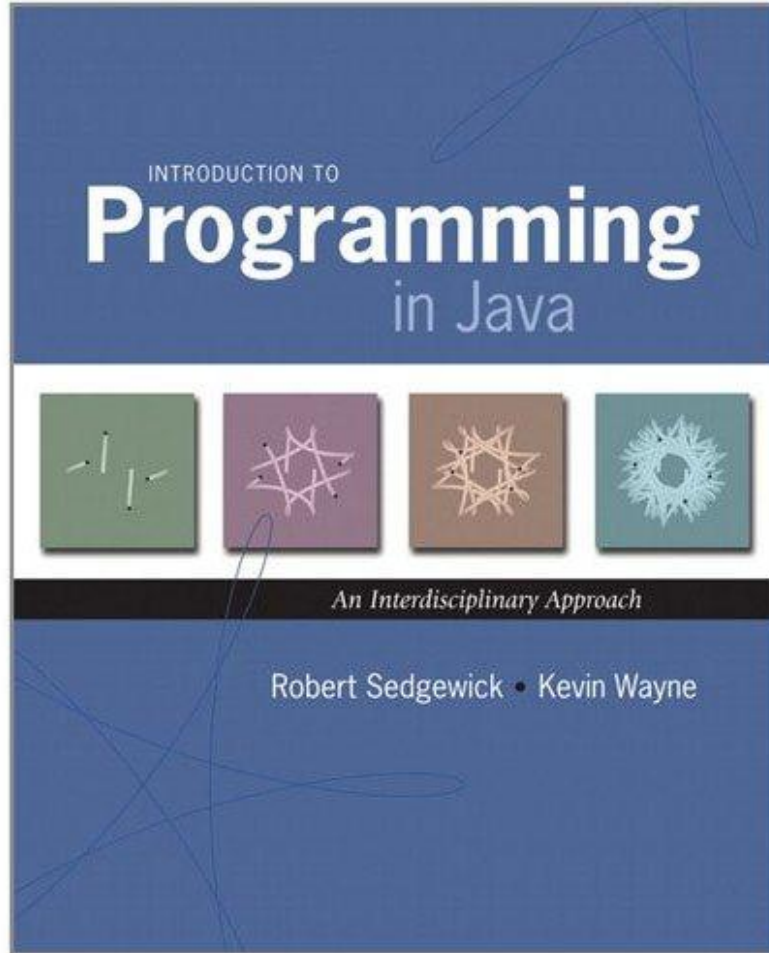
A screenshot of the Eclipse IDE interface. The main editor window displays the source code for 'HelloWorld.java'. The code includes a class declaration and a main method that prints 'Hello world!'. The Package Explorer on the left shows the project structure with 'Assignment0' containing 'src' and several Java files. The Console window at the bottom shows the output of the program: 'Hello world!'.

```
1 // Name      : Keith Vertanen
2 // Username  : kvertanen
3 // Description: My very first attempt at a Java program!
4 public class HelloWorld
5 {
6     public static void main(String [] args)
7     {
8         System.out.println("Hello world!");
9     }
10 }
11
```

<terminated> HelloWorld (1) [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Aug 14, 2011 12:12:10 PM)
Hello world!

Keith Vertanen

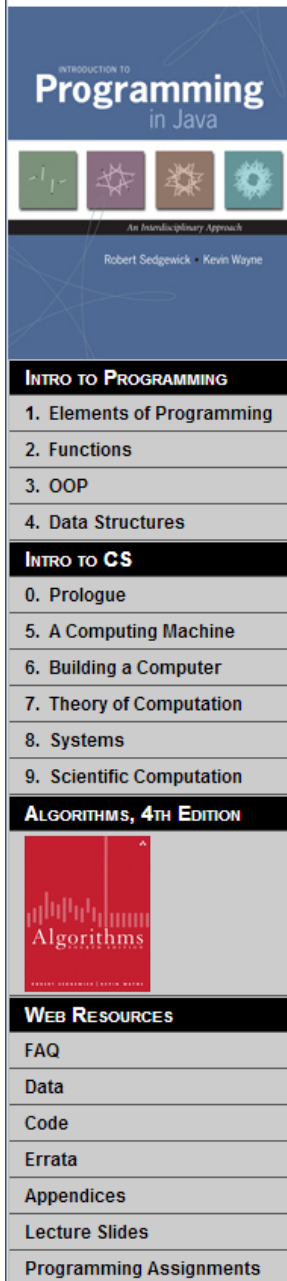
The Book



The Booksite

INTRODUCTION TO PROGRAMMING IN JAVA

*a textbook for a first course in computer science
for the next generation
of scientists and engineers*



Textbook. Our textbook *Introduction to Programming in Java* [[Amazon](#) · [Addison-Wesley](#)] is an interdisciplinary approach to the traditional CS1 curriculum. We teach all of the classic elements of programming, using an "objects-in-the-middle" approach that emphasizes data abstraction. A key feature of the book is the manner in which we motivate each programming concept by examining its impact on specific applications, taken from fields ranging from materials science to genomics to astrophysics to internet commerce. The book is organized around four stages of learning to program:

- *Chapter 1: Elements of Programming* introduces variables; assignment statements; built-in types of data; conditionals and loops; arrays; and input/output, including graphics and sound.
- *Chapter 2: Functions* introduces modular programming. We stress the fundamental idea of dividing a program into components that can be independently debugged, maintained, and reused.
- *Chapter 3: Object-Oriented Programming* introduces data abstraction. We emphasize the concept of a data type and its implementation using Java's class mechanism.
- *Chapter 4: Algorithms and Data Structures* introduces classical algorithms for sorting and searching, and fundamental data structures, including stacks, queues, and symbol tables.

Booksite. Reading a book and surfing the web are two different activities: This booksite is intended for your use while online (for example, while programming and while browsing the web); the textbook is for your use when initially learning new material and when reinforcing your understanding of that material (for example, when reviewing for an exam). The booksite consists of the following elements:

- *Excerpts.* A condensed version of the text narrative for reference while online.
- *Exercises.* Hundreds of exercises and some solutions.
- *Java code.* Hundreds of easily downloadable [Java programs](#) and [real-world data sets](#).

To get started. Here are instructions for installing a Java programming environment [[Mac OS X](#) · [Windows](#) · [Linux](#)]. We also provide [I/O libraries](#) for reading and writing text and binary data, drawing graphics, and producing sound.

To adopt. Here are some of the [distinctive features](#) of our textbook and a [marketing flyer](#). To preview our material, you can download the [preface and Chapter 1](#). If you wish to consider adoption, please fill out [this form](#) to request a copy of the textbook or ask for more information.

Last modified on January 21, 2013.

Copyright © 2002–2012 Robert Sedgewick and Kevin Wayne. All rights reserved.

<http://introc.cs.princeton.edu/java/home/>

Why learn to program?

Lots of existing software:



Knot Guide

Learn the ropes.

Sail a boat, make your own jewelry, or tie a Christmas tree to the top of your car — Knot Guide can help. Watch and learn with picture and video tutorials for 88 different types of knots.

[Learn more](#) ▶



MyNature Animal Tracks

Watch their step.

Download MyNature Animal Tracks and find out if those paw marks belong to a bobcat or a grizzly bear. Digital photos, illustrations, and range maps for over 40 animals — big and small — help you identify tracks based on size and shape.

[Learn more](#) ▶



Weber's On the Grill

Master the grill.

Get ready for your next cookout with Weber's On the Grill. Search over 250 classic Weber recipes plus 40 recipes for rubs, marinades, and sauces. Tag your favorites and tap to turn ingredients into an itemized grocery list. Before you start grilling, read through expert tips and watch instructional videos from celebrity chef Jamie Purviance.

[Learn more](#) ▶



Yoga Stretch

Strike a pose.

With Yoga Stretch, you can achieve perfect physical and mental balance without leaving home. Choose a meditative background sound and follow along with voice prompts and images for each pose. Try a preloaded routine or create a custom session that fits your body and your skill level.

[Learn more](#) ▶

...and 499,996 more and that's just iPhone apps

Reasons to program

Well...

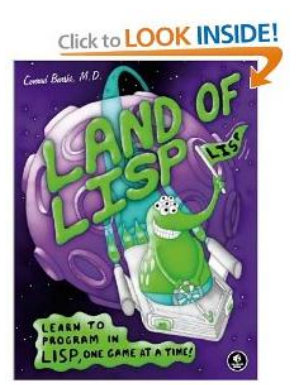
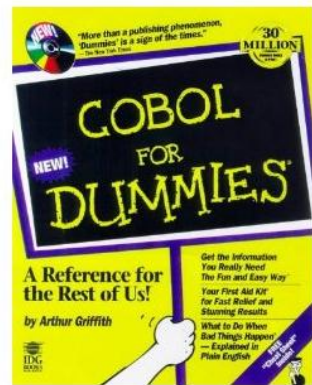
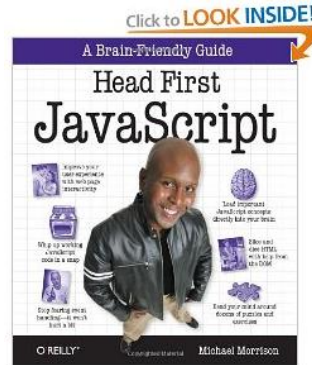
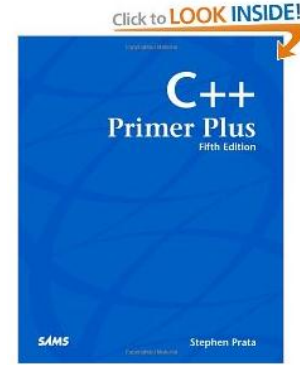
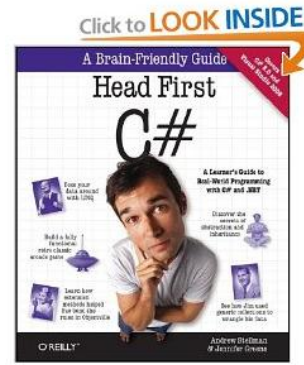
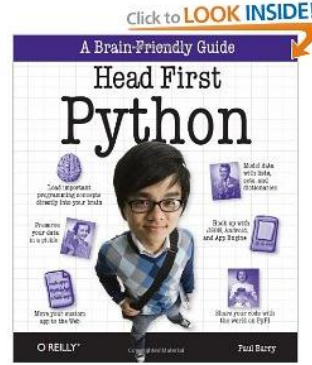
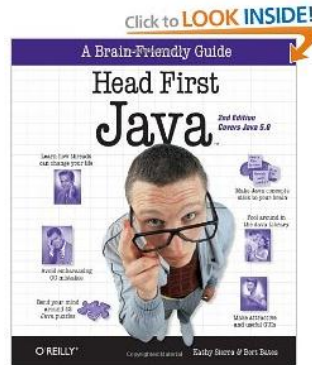
- 1 Someone had to program all those iPhone apps
(and rake in the sweet sweet profits)
- 2 Many problems are so specific to your
company/problem nobody has an app for that
- 3 Programming is fun, creative and a challenge.
- 4 Enables you to make your computer do (almost)
anything you want.

Languages

- Machine language
 - Low level, what the hardware understands
 - Tedious and error-prone to write
 - Specific to a particular type of computer
- Natural language
 - Imprecise and ambiguous
 - Hard to convert to instructions for the hardware
- High level programming language
 - Good balance between the two extremes

Becoming a programmer: step 1

Choose a language...

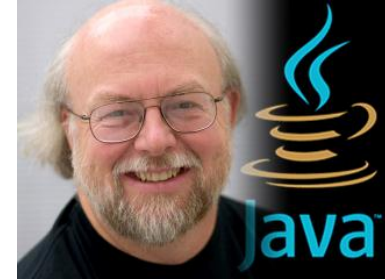


and hundreds more...

Our choice: Java

- Advantages

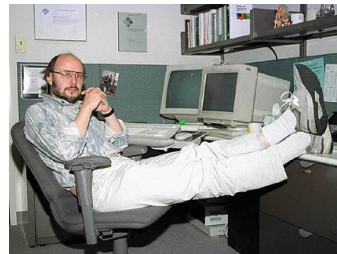
- Widely used
- Freely available
- Features help novices learn to program



James Gosling, father of Java.

- No perfect single language

- You'll learn many other languages
 - C/C++, assembly, Python, C#, JavaScript, PHP...
- Programming skills translate easily between them



*"There are only two kinds of languages: the ones people complain about and the ones nobody uses."
-Bjarne Stroustrup, father of C++*

Your first program



http://www.zazzle.com/baby_girls_first_java_program_hello_world_tshirt-235063563751392326 \$23.95

How Java works

Source code:

Plain text file created in some editor (notepad, vi, TextEdit, Eclipse, DrJava, ...)

```
public class HelloWorld
{
    public static void main(String [] args)
    {
        System.out.println("Hello world!");
    }
}
```

HelloWorld.java

% javac HelloWorld.java



Java bytecode:

Intermediate language that any device running Java can understand (humans generally ignore this)

```
Compiled from "HelloWorld.java"
public class HelloWorld extends java.lang.Object{
public HelloWorld();
    Code:
    0:      aload_0
    1:      invokespecial #1; //Method java/lang/Object."<init>":()V
    4:      return
public static void main(java.lang.String[]);
    Code:
    0:      getstatic      #2; //Field
java/lang/System.out:Ljava/io/PrintStream;
    3:      ldc          #3; //String Hello world!
    5:      invokevirtual #4; //Method
java/io/PrintStream.println:(Ljava/lang/String;)V
    8:      return
}
```

HelloWorld.class

How Java works

Java bytecode:

Intermediate language that any device running Java can understand (humans generally ignore this)

```
Compiled from "HelloWorld.java"
public class HelloWorld extends java.lang.Object{
public HelloWorld();
  Code:
    0:          aload_0
    1:          invokespecial  #1; //Method java/lang/Object."<init>":()V
    4:          return
public static void main(java.lang.String[]);
  Code:
    0:          getstatic      #2; //Field
java/lang/System.out:Ljava/io/PrintStream;
    3:          ldc          #3; //String Hello world!
    5:          invokevirtual #4; //Method
java/io/PrintStream.println:(Ljava/lang/String;)V
    8:          return
}
```

HelloWorld.class

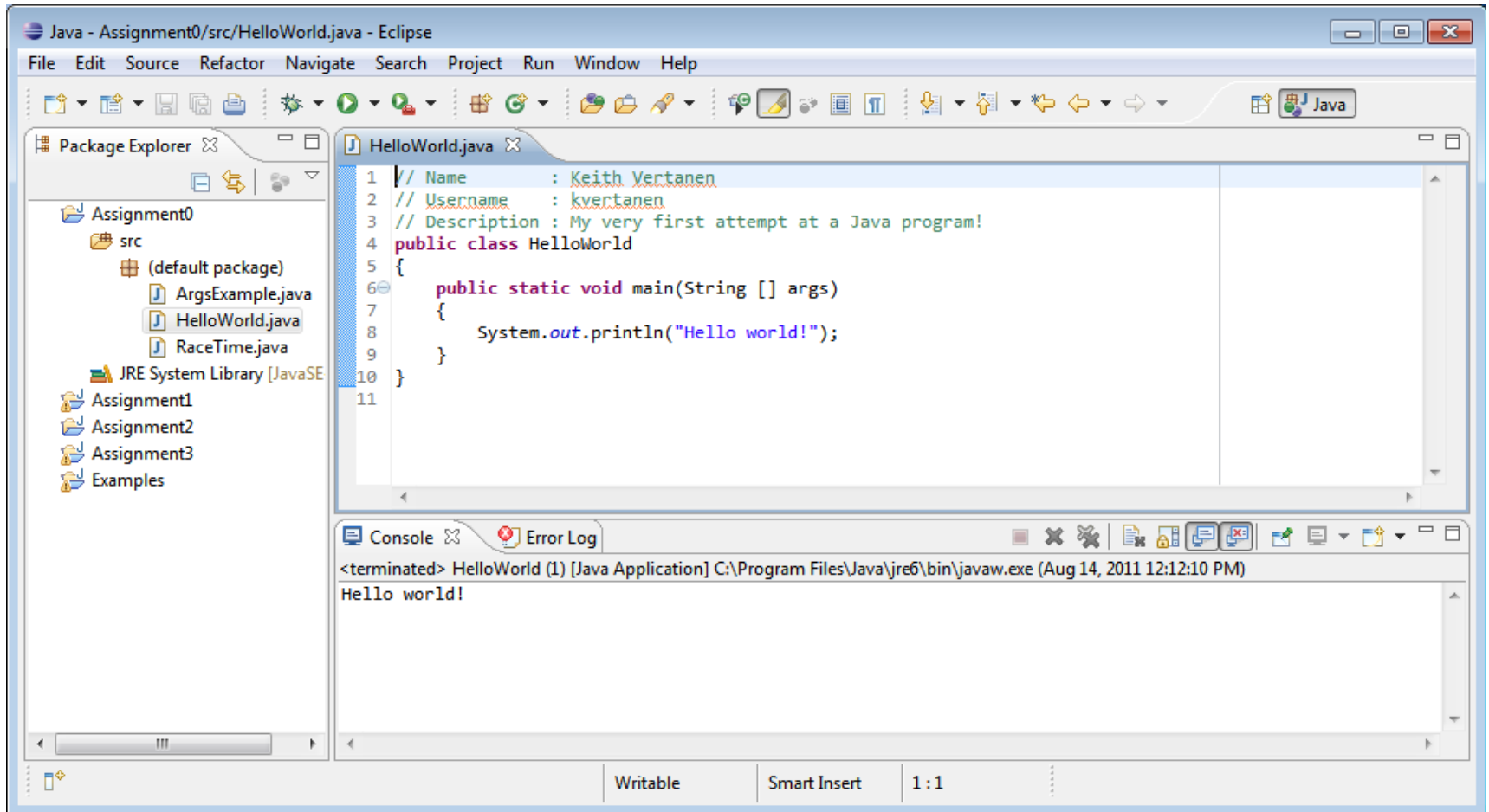
% java HelloWorld



```
Administrator: cmd
c:\Users\keith\workspace\Assignment0\src>java HelloWorld
Hello world!
c:\Users\keith\workspace\Assignment0\src>
```

Eclipse

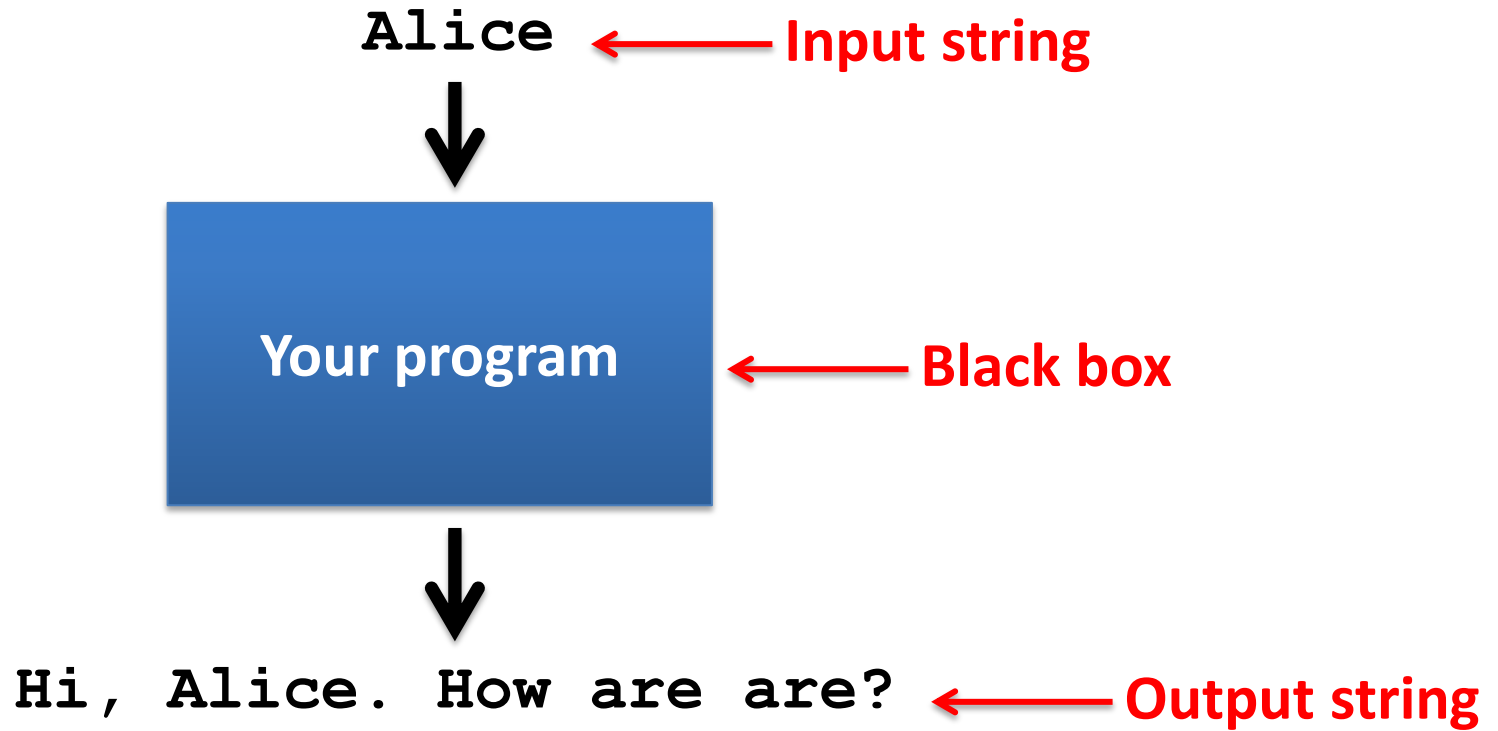
- Eclipse IDE (integrated development environment)



Eclipse

- **Eclipse IDE** (integrated development environment)
 - **Recommended** but not required
 - **Free**
 - Helpful features:
 - **Syntax highlighting**
 - Flagging likely mistakes
 - We will use mostly as a **text editor**
 - Ignoring 95% of its features
 - How to install?
 - See course web site, **resources page**
 - We'll still learn to work on the **command line**

View of programming from 10,000 feet



Anatomy of a Java program

Name of the class, must be in file named CostCalc.java (case sensitive!)

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

All the action goes here (for now)

Extra things from the command line

Allows program's **output** to depend on its **input**

```
% java CostCalc bananas 12 0.21  
To buy 12 bananas you will need $2.52
```

args array

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

```
% java CostCalc bananas 12 0.21  
To buy 12 bananas you will need $2.52
```

identifier	meaning	value	type
args[0]	1 st thing on command line after Java class name	"bananas"	String
args[1]	2 nd thing on command line	"12"	String
args[2]	3 rd thing on command line after Java class	"0.21"	String
args.length	# of things on command line	3	int

Command line args in Eclipse

The screenshot displays the Eclipse IDE interface. The Package Explorer on the left shows a project named 'Assignment0' with a source folder 'src' containing files 'ArgsExample.java', 'HelloWorld.java', and 'RaceTime.java'. The main editor shows the code for 'HelloWorld.java', which includes a 'public class' definition and a 'main' method. The Run menu is open, showing options like 'Run', 'Debug', 'Run History', 'Run As', 'Run Configurations...', 'Debug History', 'Debug As', and 'Debug Configurations...'. The 'Run Configurations' dialog is open in the foreground, titled 'Create, manage, and run configurations'. It shows a list of configurations on the left, with 'Main' selected. The 'Program arguments' field contains 'apples 6 foo'. The 'VM arguments' field is empty. The 'Working directory' is set to 'Default' with the value '\${workspace_loc:Assignment0}'. The 'Run' button is highlighted at the bottom of the dialog.

Java - Assignment0/src/ArgsExample.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Run Ctrl+F11
Debug F11

Run History
Run As
Run Configurations...
Debug History
Debug As
Debug Configurations...

Toggle Breakpoint
Toggle Line Breakpoint
Toggle Method Breakpoint
Toggle Watchpoint
Skip All Breakpoints
Remove All Breakpoints
Add Java Exception Breakpoint...
Add Class Load Breakpoint...

All References...
All Instances...
Instance Count...
Watch
Inspect
Display
Execute
Force Return

Run Configurations
Create, manage, and run configurations
Run a Java application

Name: ArgsExample

Main (*)= Arguments JRE Classpath Source Environment Common

Program arguments:
apples 6 foo
Variables...

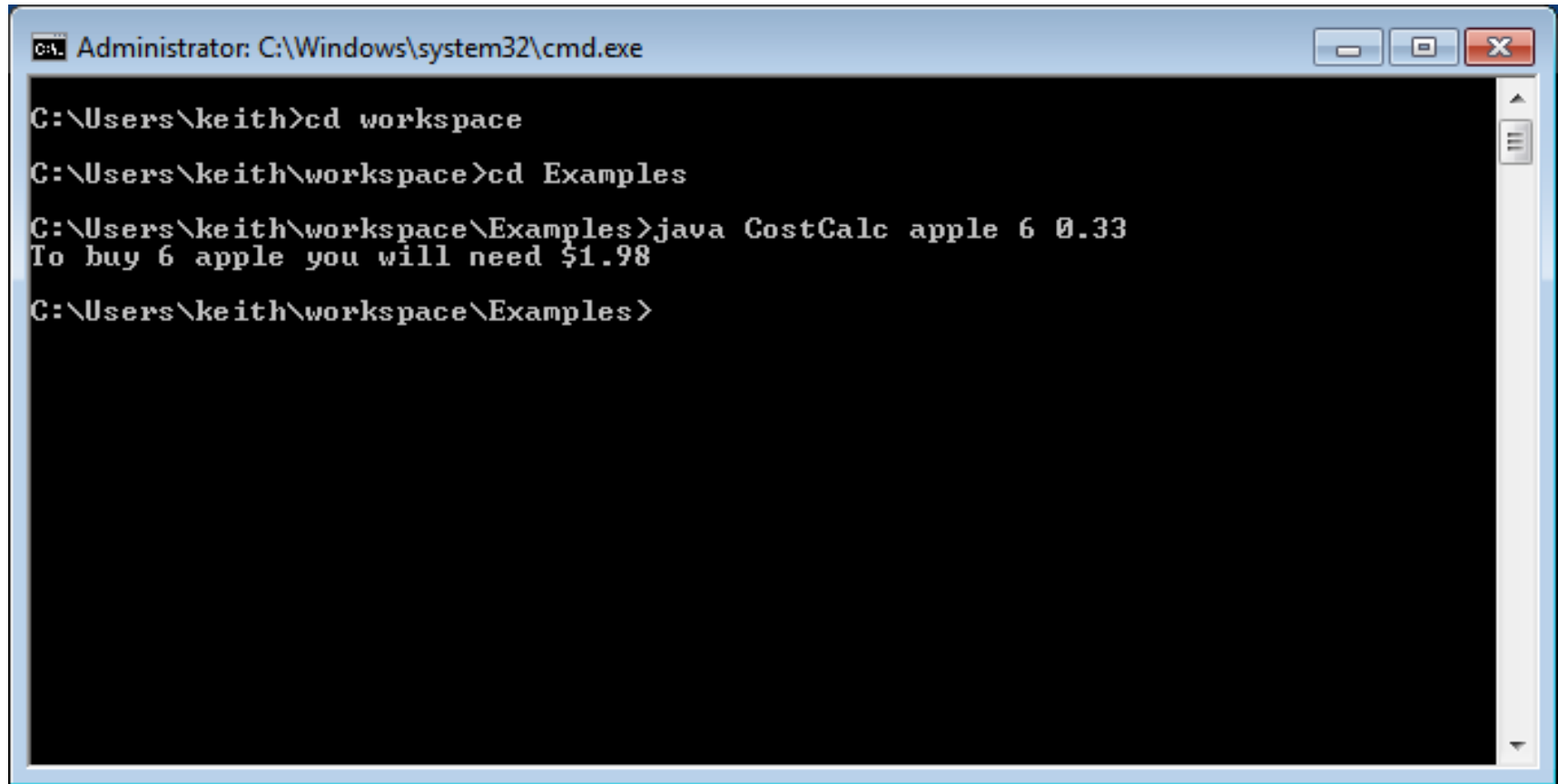
VM arguments:
Variables...

Working directory:
Default: \${workspace_loc:Assignment0}
Other:
Workspace... File System... Variables...

Filter matched 22 of 25 items

Run Close

Command line args in command shell



```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\keith>cd workspace
C:\Users\keith\workspace>cd Examples
C:\Users\keith\workspace\Examples>java CostCalc apple 6 0.33
To buy 6 apple you will need $1.98
C:\Users\keith\workspace\Examples>
```

Summary

- Source code → byte code → program output
- Wrote our first program
 - Hello world!
- Program have **input**
 - e.g. arguments passed into `main()`
- Programs have **output**
 - e.g. text printed from `System.out.println()`