Chapter 1

How to get started with Java and NetBeans

Objectives

Applied

- Given a NetBeans project that contains the source code for a Java application, use NetBeans to open the project, view and compile the source code, and run the application.
- Given the source code for a Java application, use NetBeans to create a project; enter edit, and compile the source code; and run the application.

Objectives (cont.)

Knowledge

- Describe how Java compares with C++ and C# based on these features: syntax, platform independence, speed, and memory management.
- Name and describe the three types of programs that you can create with Java.
- Describe how Java compiles and interprets code.
- Explain how the use of bytecodes lets Java achieve platform independence.
- Describe the benefits of using a Java IDE like NetBeans or Eclipse.
- Describe how NetBeans detects and displays syntax errors.

Java timeline

Year	Release/Event
1996	Java Development Kit 1.0 (JDK 1.0).
1997	Java Development Kit 1.1 (JDK 1.1).
1998	Java 2 Platform with version 1.2 of the Software Development Kit (SDK 1.2).
1999	Java 2 Platform, Standard Edition (J2SE).
	Java 2 Platform, Enterprise Edition (J2EE).
2000	J2SE with version 1.3 of the SDK.
2002	J2SE with version 1.4 of the SDK.
2004	J2SE 5.0 with version 1.5 of the JDK.
2006	Java SE 6 with version 1.6 of the JDK.
2010	Oracle buys Sun.
2011	Java SE 7 with version 1.7 of the JDK.

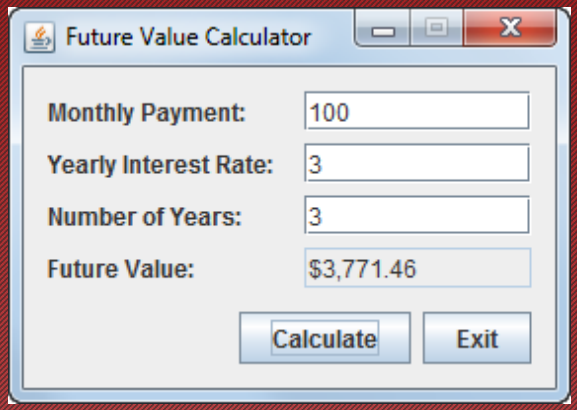
Operating systems supported by Java

- Windows (XP, Vista, 7)
- Linux
- Solaris
- Macintosh OS X

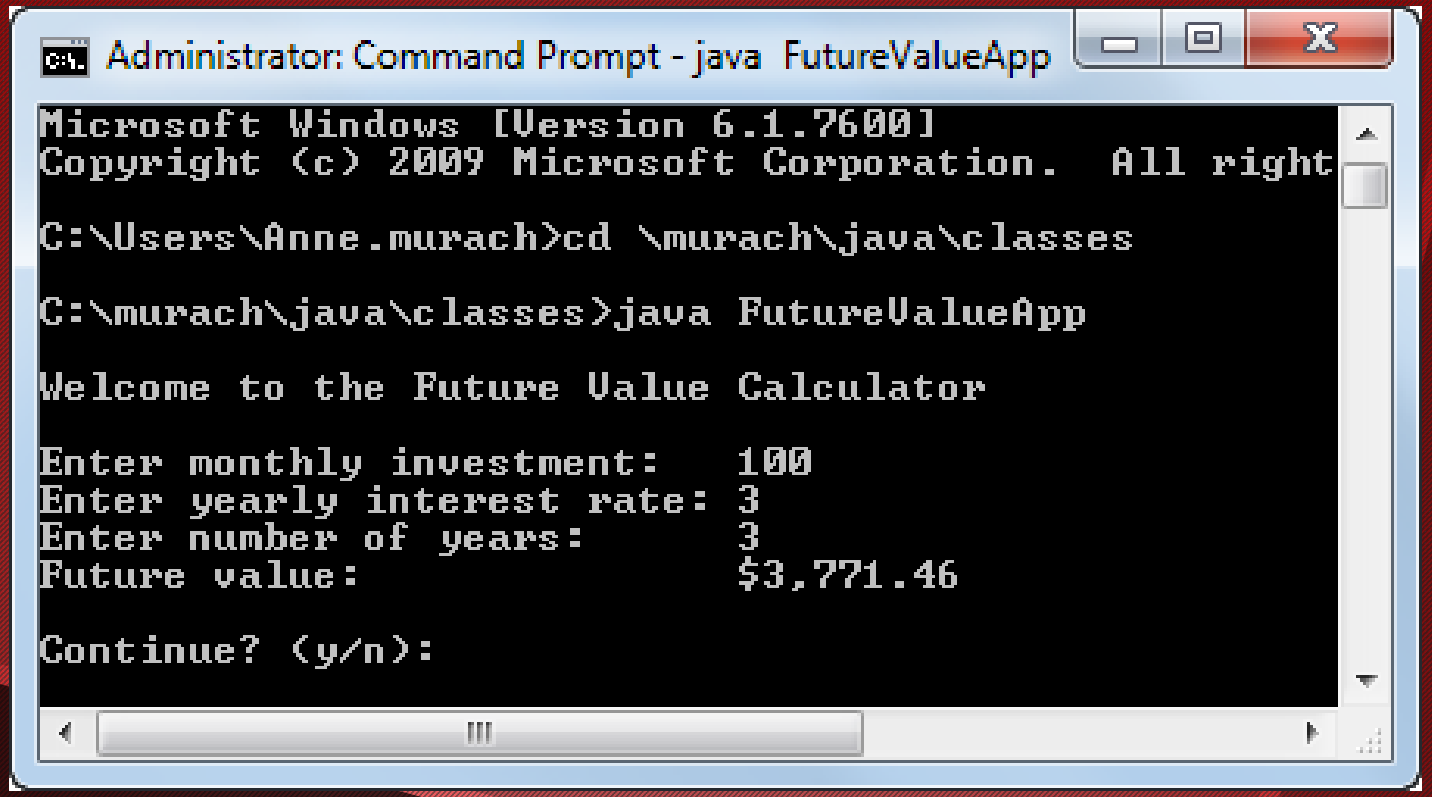
Java compared to C++ and C#

Feature	Description
Syntax	Java syntax is similar to C++ and C# syntax.
Platforms	Compiled Java code can be run on any platform that has a Java interpreter. Similarly, compiled C# code (MSIL) can be run on any system that has the appropriate interpreter. Currently, only Windows has an interpreter for MSIL. C++ code must be compiled once for each type of system that it is going to be run on.
Speed	C++ and C# run faster than Java, but Java is getting faster with each new version.
Memory	Both Java and C# handle most memory operations automatically, while C++ programmers must write code that manages memory.

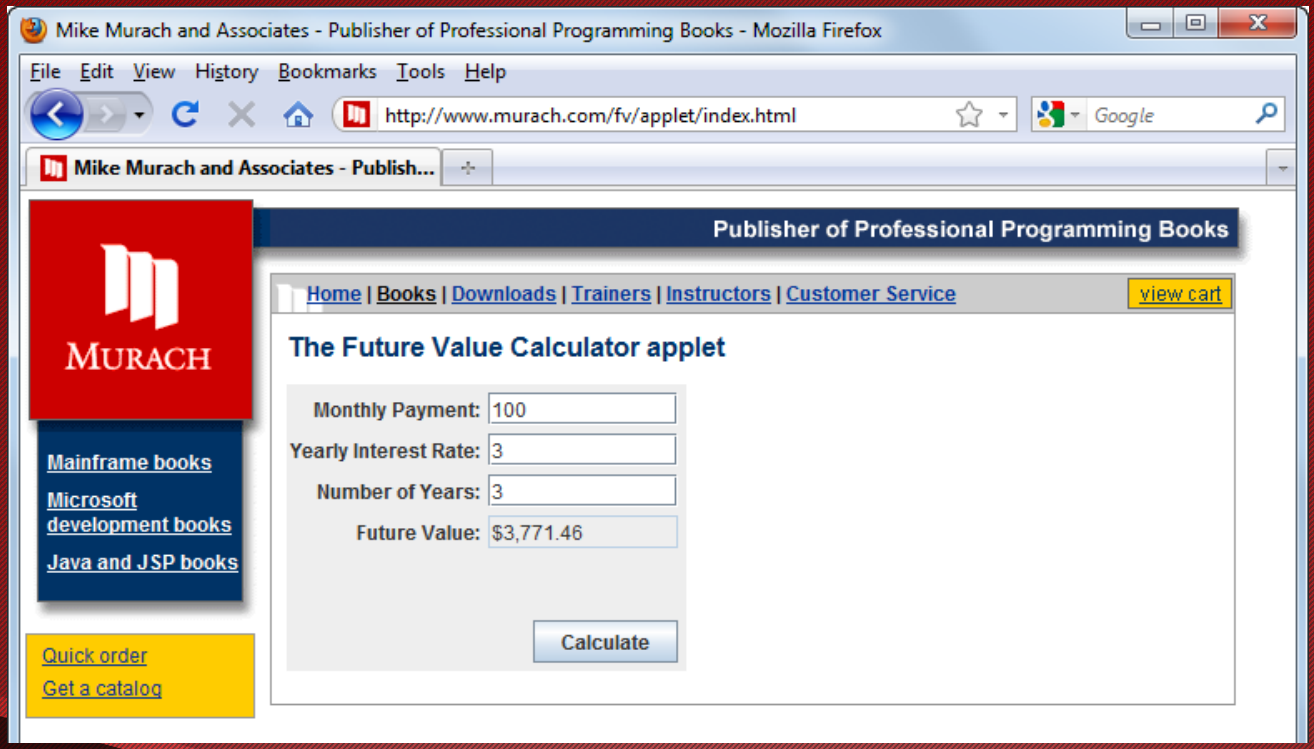
A GUI application



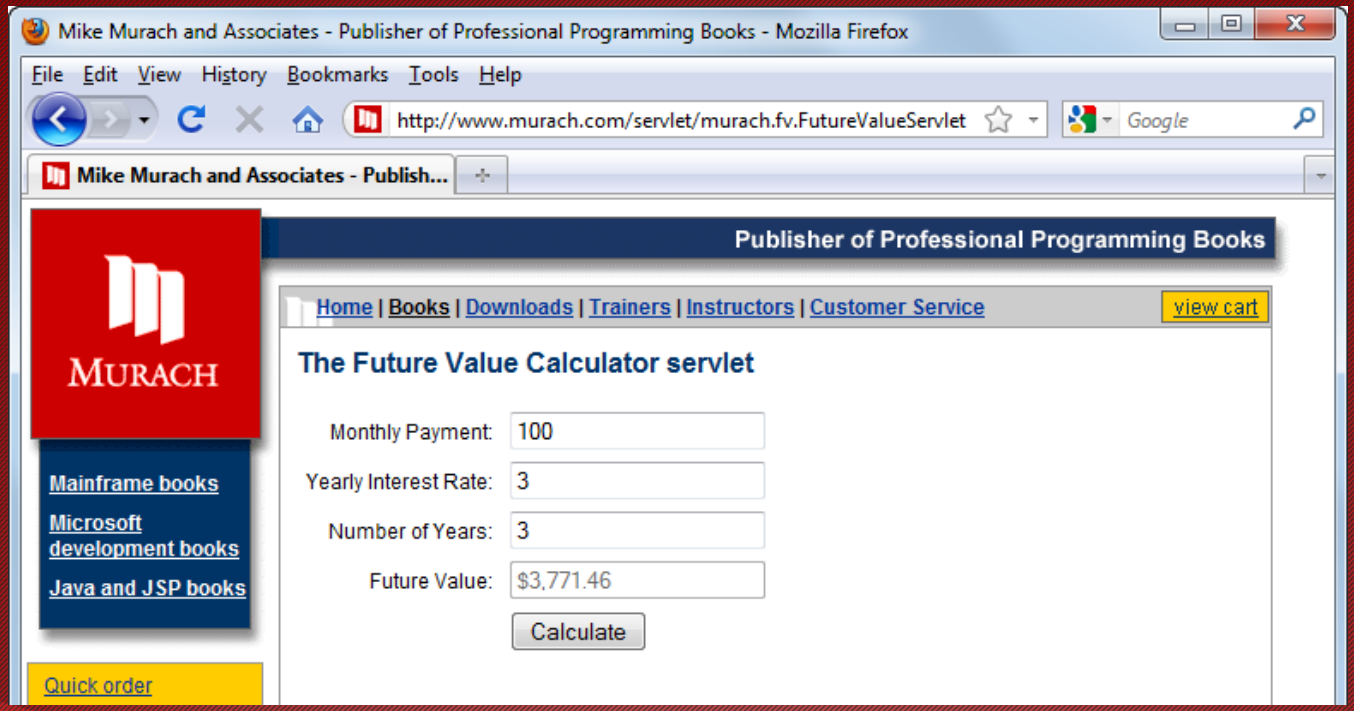
A console application



An applet



A servlet



The code for the Future Value application

```
import java.util.Scanner;
import java.text.NumberFormat;

public class FutureValueApp
{
    public static void main(String[] args)
    {
        System.out.println(
            "\nWelcome to the Future Value Calculator\n");

        Scanner sc = new Scanner(System.in);
        String choice = "y";

        while (choice.equalsIgnoreCase("y"))
        {
            // get the input from the user
            System.out.print(
                "Enter monthly investment:   ");
            double monthlyInvestment = sc.nextDouble();
            System.out.print(
                "Enter yearly interest rate: ");
```

The code for the Future Value application (cont.)

```
double interestRate = sc.nextDouble();
System.out.print(
    "Enter number of years:      ");
int years = sc.nextInt();

// calculate the future value
double monthlyInterestRate =
    interestRate/12/100;
int months = years * 12;
double futureValue = calculateFutureValue(
    monthlyInvestment, monthlyInterestRate,
    months);

// format and display the result
NumberFormat currency =
     NumberFormat.getCurrencyInstance();
System.out.println(
     "Future value:                " +
     currency.format(futureValue) + "\n");
```

The code for the Future Value application (cont.)

```
            // see if the user wants to continue
            System.out.print("Continue? (y/n): ");
            choice = sc.next();
            System.out.println();
        }
    }

    private static double calculateFutureValue(
    double monthlyInvestment,
    double monthlyInterestRate, int months)
    {
        double futureValue = 0;
        for (int i = 1; i <= months; i++)
            futureValue =
            (futureValue + monthlyInvestment) *
            (1 + monthlyInterestRate);
        return futureValue;
    }
}
```

How Java compiles and interprets code