Programming in Java

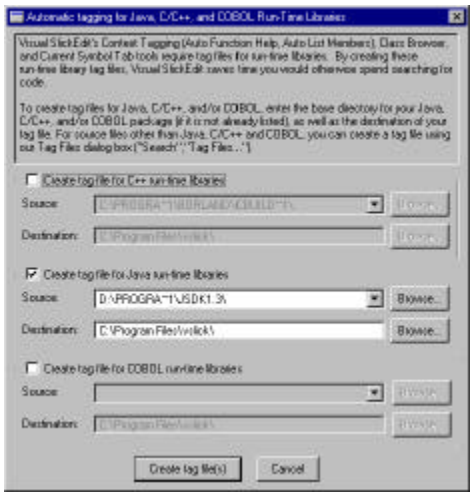
Day 2

Review of Compiling Issues

- Be sure path is set in autoexec.bat
- c:\jdk1.3\bin
- In Windows 95 this should be
  - in the file c:\autoexec.bat
  - edit this file with Vslick
  - to contain
  - `set path=%path%;c:\jdk1.3\bin`

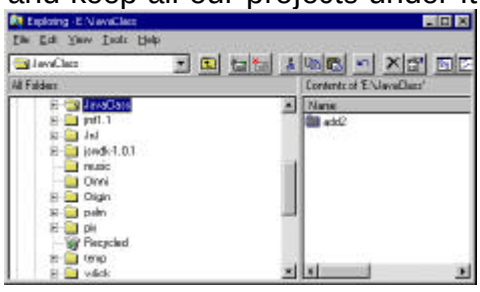
Tag Files allow Syntax Completion

- Search | Tag files |
- Auto tag



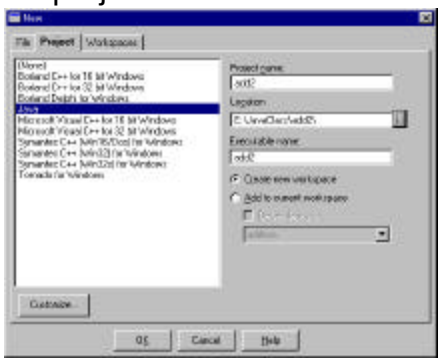
Setting up a Project in VSlick

- Particularly advantageous for multiple files.
- First create a folder for the project
- Lets create a folder called JavaClass
  - and keep all our projects under it



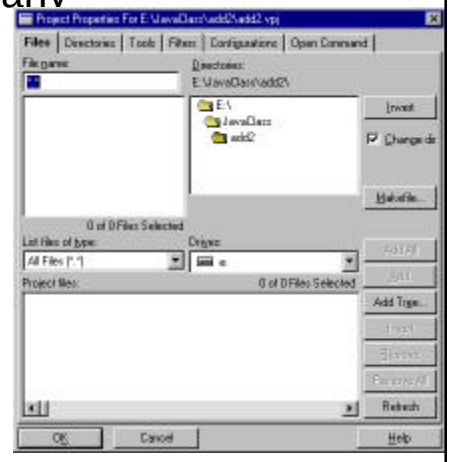
Then Create the Project

- Project|New
- Give the project the same name as the main program file



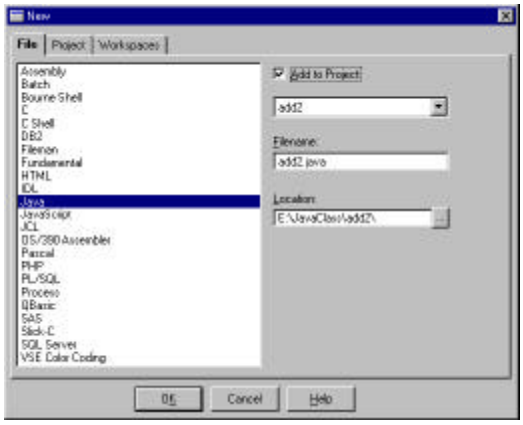
The next screen asks for files to add to the project

- Initially there won't be any
- Just close the window



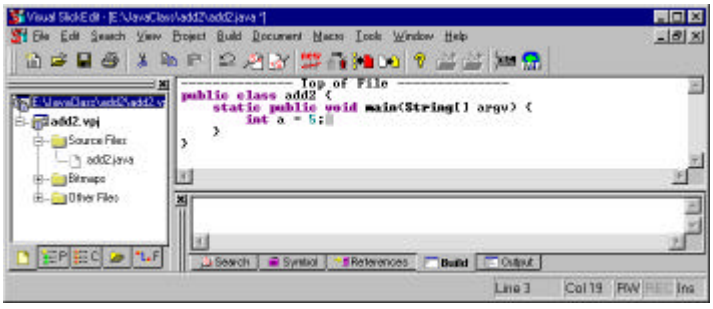
Then create the first file

- File|New



And begin typing in code

- Note file list in project on left

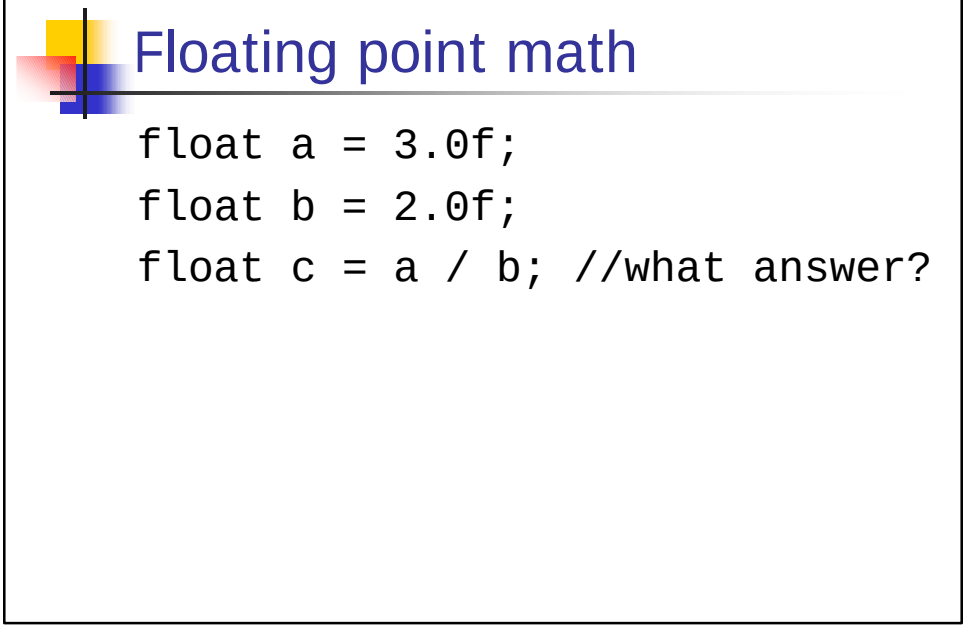


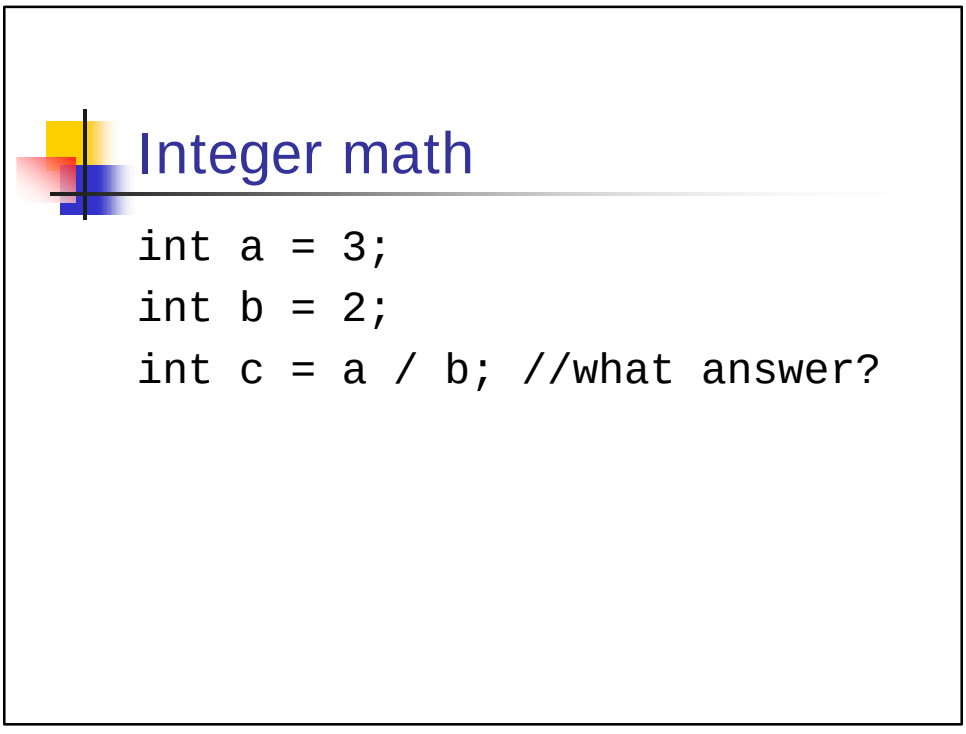
To compile the program

- Save with File|save or Ctrl/S
- Build with Build|Build or Ctrl/M
- Run program with Build|Execute
  - or Alt/B x

Arithmetic Operations in Java

+	add
-	subtract
*	multiply
/	divide
%	modulo (remainder after division)


Floating point math
float a = 3.0f;
float b = 2.0f;
float c = a / b; //what answer?


Integer math
int a = 3;
int b = 2;
int c = a / b; //what answer?

Modulo or remainder

```
int a = 5;
int b = 3;
int quotient = a / b;  //what?
int remainder = a % b; //what?
```

Mixed math

```
float a = 3.0f;
int b = 2;
float c = a / b;
System.out.println(“c=“ + c);
```

Incrementing and Decrementing

```
j = j + 1;
j++;
j = j-1;
j--;
x = x + 5;
x += 5;
y -= 3;
```

Making decisions

```
if (x  < 12)
  System.out.println(“too small”);

  if (z >= 132)
    System.out.println(“z big”);
  else
    System.out.println(“z small”);
```

Better to always use braces

```
if (x >= 273.16) {
  System.out.println(“>0”);
}
else {
  System.out.println(“<=0”);
}
```

A real program

```
public class SimpleIf {
    static public void main(String argv[]) {
        float a = 3.4f ;
        float b = 1.2f ;
        float c = a / b;
        if (c < 2 ) {
            System.out.println("c < 2 : "+ c);
        }
        else {
            System.out.println("c >= 2: "+c);
        }
    }
}
```

While loop

```
i = 0;
while (i < 10) {
  System.out.println(i++);
}
```

Notes on while

- Test made at top of loop.
- May never be executed.
- Must initialize variables before loop starts.
- Can write loops that never exit.

```
i = 0;
while (i < 10) {
   System.out.println(i++);
}
```

- Consider

```
i = 0;
while (i < 10) {
   System.out.println(i);
}
```

for Looping

```
for (i =0; i < 10; i++) {
  System.out.println(i + ” “ + i*5);
}
```

Details on for loop

- Fixed number of times through loop
- Three arguments
  - initial value
  - terminating condition
  - action to take after each pass through loop
- Loop may never be executed
- Test is made at top of loop.
- Possible to write loops that do not exit.

```
for (i =0; i < 10; i++) {
   System.out.println(i);
}
```

do-while

```
i = 0;
do {
  System.out.println(i++);
}
while (i < 10);
```

Notes on do-while

- Test made at *bottom* of loop.
- Always executed once.
- Can write loops that never exit.
- Least frequently used loop
  - Hardest to read.

```
i = 0;
do {
   System.out.println(i++);
}
while (i < 10);
```

All the loops at once

```
public class AlltheLoops {
    static public void main(String argv[] ) {
        int i = 0;
        while (i < 10) {
            System.out.print(i++ +" ");
        }
        System.out.println();
        for(i = 0; i < 10; i++ ) {
            System.out.print(i++ +" ");
        }
        System.out.println();
        i=0;
        do {
            System.out.print(i++ +" ");
        } while (i < 10 );
        System.out.println();
    }
}
```

- Can you spot the mistake?

Subroutines in Java

- Are called functions
  - Can have return values or not
  - arguments or not

```
public void whileLoop() {
        int i = 0;
        while (i < 10) {
            System.out.print(i++ +" ");
        }
        System.out.println();
}
```

The while loop as a function

```
public void whileLoop() {
        int i = 0;
        while (i < 10) {
            System.out.print(i++ +" ");
        }
        System.out.println();
    }
```

The for loop as a function

```
public void forLoop() {
        for (int i = 0; i < 10; i++ ) {
            System.out.print(i +" ");
        }
        System.out.println();
    }
```

doWhile function

```
public void doWhile() {
        int i=0;
        do {
            System.out.print(i++ +" ");
        } while (i < 10 );
        System.out.println();
    }
```

Then we could call them

```
whileLoop();
forLoop();
doWhile();
```

You might think that you could call these from the main routine

```
static public void main(String[] argv) {
  whileLoop();
  forLoop();
  doWhile();
}
```

- But this is not legal in Java
- The only function you can call from the static main routine is one to create an instance of the class.

```
new FunctionLoops();
```

The reason for this is

- Every Java module is a class
- Each class can have one or more instances
  - each having its own values for internal variables
- Only the static main routine has no instances.
- We have to create at least one instance of these classes.

Best way to write Java classes

```
public class foo {
   public foo() {
      //code goes here
   }
   static public void main(String[] argv) {
      new foo();
   }
}
```

For our loop function program

```
public class FunctionIfs {
//---------this is the constructor--------
    public FunctionIfs() {
        whileLoop();
        forLoop();
        doWhile();
    }
//-----------------
static public void main(String[] argv) {
  new FunctionIfs();
}
```

Let's consider a simple Window

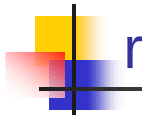
- Windows appear on the screen with borders.
- A Frame is a more elaborate Window
  - with a title bar.
- Our Rect1 class will draw rectangles in a Frame window.

```
public class Rect1 extends Frame
```

All windowing programs use

- the AWT
  - Advanced Windowing Toolkit library
  - We import this to make these functions available

```
import java.awt.*;
```



Since windows need redrawing

- We must provide a paint routine that is called when
  - Windows are resized
  - maximized
  - hidden and uncovered
- It redraws the interior of the window

```
public void paint(Graphics g) {
}
```

We will have two classes

- Rect1
  - contains main()
  - creates two instances of Rectangl class
- Rectangl
  - does drawing

Now lets consider the Rectangl class

```
import java.awt.*;
public class Rectangl {
    private int xpos, ypos;
    private int width, height;

    public Rectangl(int x, int y, int w, int h) {
        xpos = x;     //remember size and posn
        ypos = y;
        width = w;
        height = h;
    }
//-------------------------------------------
    public void draw(Graphics g)      {
    //draws rectangle at current position
        g.setColor(Color.blue);
        g.drawRect(xpos, ypos, width,height);
    }
}
```

And the calling program

```
import java.awt.*;
public class Rect1 extends Frame {
    private Rectangl rect1;             //two rectangle objects
    private Rectangl rect2;
//-------------------------------------------
public Rect1() {

   super("Rect1 window");  //create window with title bar
   //Create rectangles and tell them where to draw
        rect1 = new Rectangl(10, 40, 200, 100);
        rect2 = new Rectangl(70, 50, 150, 75);
        setBounds(50, 50, 475, 225);      //size of window
        setVisible(true);                 //display window
 }
```

Rest of Rect drawing

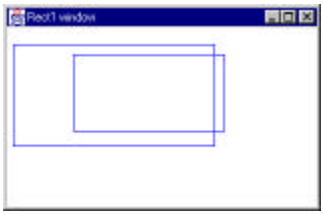
```
//-----------------------------------------
    public void paint(Graphics g) {
        rect1.draw(g);
        rect2.draw(g);
    }
//-----------------------------------------
    public static void main(String args[]) {
        new Rect1();    //create instance of Rect1 class
    }
}
```

Major points to notice

- `import java.awt.*;`
  - Means get function definitions from java.awt library (called a package)

```
public class Rect1 extends Frame {
    private Rectangl rect1;
    private Rectangl rect2;
```

Resulting program window



So far we've considered:

- Computers and compilers
- Java interpreters
- Java language syntax
- Java variable types
- Basics of compiling
- Looping constructs
- Functions
- Drawing rectangles in windows

Homework Assignment

- Write a program to print out the squares of the even numbers between 2 and 20
- Have a great holiday