Running And Compiling Java

From the Command Prompt
The Basic Approach

- Nice tools like Eclipse and JBuilder can assist compilation, debugging and running.

- You must know the basic approach which does not require a tool!
  - Uses the Command Prompt

- I’ll assume that you already have written code called Awesome.java
Step 1: Getting a Prompt

- Start the Command Prompt tool
Step 2: Go To Directory

- Your code will be in some folder.
  - **Examples:**
    - C:\Awesome.java
    - C:\NiftyCodeFolder\Awesome.java
    - C:\Documents and Settings\AllisonSmith\My Documents\Java_Code\Awesome.java
    - C:\Documents and Settings\LisaSimpson\My Documents\homework\Awesome.java

- Command Prompt should be in that folder
  - type “cd ” with a space, followed by the proper folder
  - **e.g.**
    - cd C:\NiftyCodeFolder
    - cd C:\Documents and Settings\AllisonSmith\My Documents\Java_Code
Example

- Changing directories (folders)

- Going back up one directory with “..”
Directory Manipulation

There are shortcuts for getting around directories.

- For example, don’t need to type entire path.
  - Just type the rest of the path from where you are

- Can learn about shortcuts elsewhere (e.g., online)
Step 3: Compile

- Type “javac Awesome.java”
  - Of course, if your code is called something else, substitute the correct name.
  - This creates the byte code.
    - You’ll see a new file called Awesome.class
Step 4: Run

- Type “java Awesome”
  - Runs the byte code
    - You could delete Awesome.java and just leave Awesome.class!

In my example above, the code ran and printed out a line of text.
Step 5: Errors

If your code has syntax errors, they will be printed.

• must correct your code and try again
Two Types Of Errors

- **Compilation errors** will be displayed in the Command Prompt
  - Compilation errors are mistakes in your java syntax
    - Example: `System.out.println("\"\")`; instead of `System.out.println("\"\")`

- **Runtime errors** are different and only appear when you run the code.
  - Example: division by zero

- Errors can be cryptic!
  - Takes experience to learn what the errors mean.
Common Beginner Errors

- Code does not have same name as file.
  - Awesome.java must contain a class called Awesome

```java
public class WrongName
{
    ... etc.
}
```

- Missing brackets
  - Every "{" must have a "}" and vice versa

```java
public class Awesome
{
    public static void main(String[] args)
    {
        System.out.println("This code only prints out this silly line.");
    }
}
```

missing its mate
Other Beginner Errors

- Typos
  - System.out.println(“…”);

- Wrong data types assigned to variables.
  - int i = 3.5;

- Computer’s path variable set improperly during installation.
  - See my Java Installation instructions
Step 6: Advanced Alternative! (Packages)

- Instead of “cd-ing” to the directory
  1. Can specify the path when compiling
     - `javac C:\NiftyCodeFolder\Awesome.java`

- But inside the code, **a line must be added**
  - `package NiftyCodeFolder;`
  - Must be the first line of the code!
  - Java calls folders “packages”

- Now compile with “.” in place of “\”
  - `java NiftyCodeFolder.Awesome`

- **This is useful later when we “import” other code**
  - Can store other code in meaningfully named folders.
  - Not useful all the time.
The code

```java
package NiftyCodeFolder;

public class Awesome {
    public static void main(String[] args) {
        System.out.println("This code only prints out this silly line.");
    }
}
```

The result

![Command Prompt window showing the output of the code]