

HelloWorld code discussion

What are the *variables* in this code?

Remember the formula for the area of a circle: $A = r^2$? In this formula, r represents the radius of the circle, whatever that is. Variables in computer programming work in a similar way to the symbols in a mathematical formula, like r here.

In Java, you need to include at least two things in a variable declaration:

- 1) First, Java needs to know the type of the variable (primitive types include integer type, like 0, 19, or -78, Boolean type, like true or false, character type, like 'K' or the backspace character, and float and double types, for lower and higher-precision decimal numbers). Knowing what type the variable is lets Java allocate (reserve) enough memory to hold any value of that type and tells it how to expect that variable to behave in the program. (In the area calculation, the r in the area formula is implicitly understood to be a number, rather than a Boolean value like true, or a letter like 'N'.)
- 2) Second, you need to give the variable a name, so you can refer back to it later. Java has a few rules for naming variables. You can't start names with a number, you can't put spaces in the name (use underscores `_` for this instead), and you can't use certain weird punctuation marks.

Here are the two integer-type variables you declared in HelloWorld:

```
int birthyear, age;
```

Here is the String variable you declared. String variables can hold text, as an array of characters.

```
String name;
```

Why do I have to type more stuff to read in the birthyear than I do to read in the name?

Input from the keyboard comes in the form of a text String, so that Java can easily match types with String name. But integer numbers are limited to digits, not characters, so Java has to use a *parser* to do the conversion between Strings of the characters '0', '1', '2', ..., '9' and actual integers. Parsing has to do with taking some kind of input and extracting information or meaning from it. You parse English into ideas when you read about something new and understand it. In this code example, parsing from a String ("81") to a number (81) is an important step in getting the user input to work.

In the homework, you're asked to get rid of the variables declared above and instead declare a decimal number r , for example like this.

```
double r;
```

Declare an auxiliary variable as well; we'll use this to streamline the user input:

```
Float r1;
```

Prompt the user to enter a number, like 5.1 or 2.2.

```
System.out.println("Enter a distance: ");
```

Convert the user's input (like "5.1") to a Float object (like 5.1). This conversion is useful because Float objects have a parser. Parsing a decimal number requires taking the

decimal point into account. We could write code to do this, but it's easier to use Java's own methods.

```
r1=Float.valueOf(stdin.readLine());
```

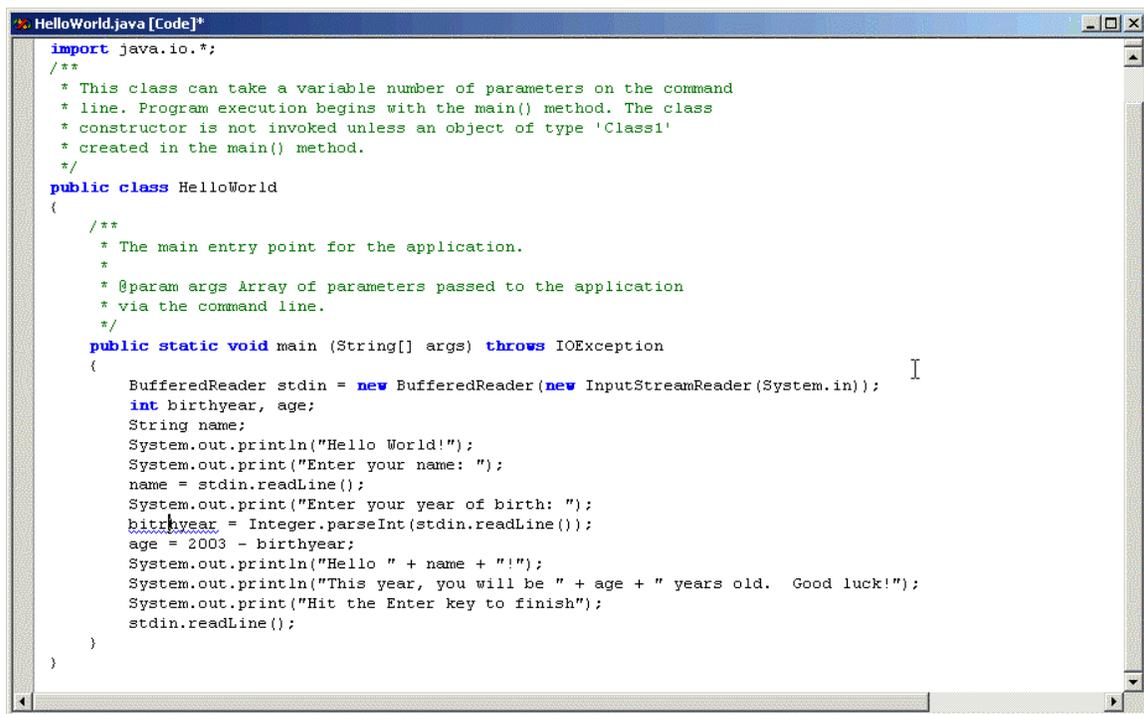
Now convert the Float value you just parsed to a double precision floating-point number.

```
r = r1.doubleValue();
```

What can go wrong in this code, when you try to compile (Build)? Compiling is the process of translating your Java code, which you can understand (we hope) into a language the computer can easily understand. The errors that turn up at this point are usually *syntax errors*. When you think about it, the compiler is one big parser, which may help you figure out your errors. Here are a few common errors:

Typos

For example, your variable name is misspelled at some place in your program.



```
import java.io.*;
/**
 * This class can take a variable number of parameters on the command
 * line. Program execution begins with the main() method. The class
 * constructor is not invoked unless an object of type 'Class1'
 * created in the main() method.
 */
public class HelloWorld
{
    /**
     * The main entry point for the application.
     *
     * @param args Array of parameters passed to the application
     * via the command line.
     */
    public static void main (String[] args) throws IOException
    {
        BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
        int birthyear, age;
        String name;
        System.out.println("Hello World!");
        System.out.print("Enter your name: ");
        name = stdin.readLine();
        System.out.print("Enter your year of birth: ");
        birthyear = Integer.parseInt(stdin.readLine());
        age = 2003 - birthyear;
        System.out.println("Hello " + name + "!");
        System.out.println("This year, you will be " + age + " years old. Good luck!");
        System.out.print("Hit the Enter key to finish");
        stdin.readLine();
    }
}
```

Task List - 1 Compile/Build/Deploy task shown			
	Description	File	Line
	Click here to add a new task		
!	Undefined name 'birthyear' (J0049)	D:\TA 1200 Spring\... \HelloWorld\HelloWorld.java	27

Or you can mistype the name of a Java-defined object.

```
>HelloWorld.java [Code]*
import java.io.*;
/**
 * This class can take a variable number of parameters on the command
 * line. Program execution begins with the main() method. The class
 * constructor is not invoked unless an object of type 'Class1'
 * created in the main() method.
 */
public class HelloWorld
{
    /**
     * The main entry point for the application.
     *
     * @param args Array of parameters passed to the application
     * via the command line.
     */
    public static void main (String[] args) throws IOException
    {
        BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
        int birthyear, age;
        String name;
        System.out.println("Hello World!");
        System.out.print("Enter your name: ");
        name = stdin.readLine();
        System.out.print("Enter your year of birth: ");
        birthyear = Integer.parseInt(stdin.readLine());
        age = 2003 - birthyear;
        System.out.println("Hello " + name + "!");
        System.out.println("This year, you will be " + age + " years old. Good luck!");
        System.out.print("Hit the Enter key to finish");
        stdin.readLine();
    }
}
```

Task List - 1 Compile/Build/Deploy task shown			
	Description	File	Line
	Click here to add a new task		
!	Undefined name 'system' (30049)	D:\TA 1200 Spring\... \HelloWorld\HelloWorld.java	29

Missing semicolons

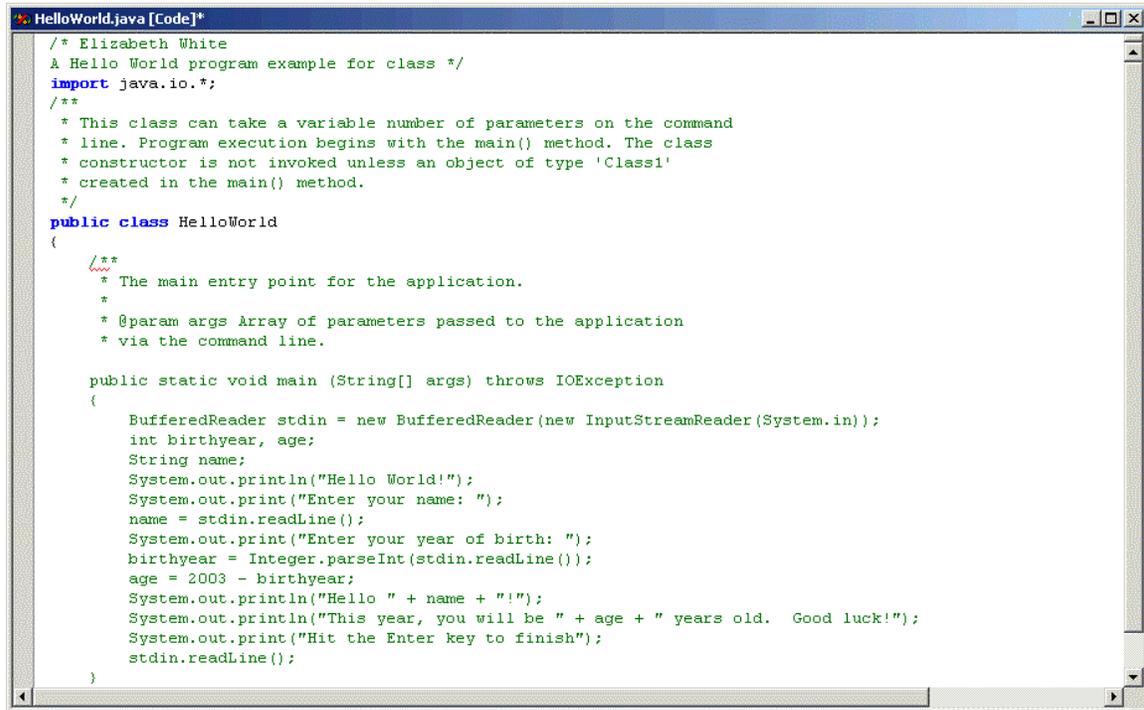
Most simple Java statements end with a ; (like a period ends a sentence in English).
Notice that the error only shows up on the next line!

```
/* Elizabeth White
A Hello World program example for class */
import java.io.*;
/**
 * This class can take a variable number of parameters on the command
 * line. Program execution begins with the main() method. The class
 * constructor is not invoked unless an object of type 'Class1'
 * created in the main() method.
 */
public class HelloWorld
{
    /**
     * The main entry point for the application.
     *
     * @param args Array of parameters passed to the application
     * via the command line.
     */
    public static void main (String[] args) throws IOException
    {
        BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
        int birthyear, age;
        String name;
        System.out.println("Hello World!");
        System.out.print("Enter your name: ")
        name = stdin.readLine();
        System.out.print("Enter your year of birth: ");
        birthyear = Integer.parseInt(stdin.readLine());
        age = 2003 - birthyear;
        System.out.println("Hello " + name + "!");
        System.out.println("This year, you will be " + age + " years old. Good luck!");
        System.out.print("Hit the Enter key to finish");
        stdin.readLine();
    }
}
```

Task List - 1 task			
I	Description	File	Line
	Click here to add a new task		
!	Expected ';'.	D:\TA 1200 Spring\...\HelloWorld\HelloWorld.java	25

Unterminated comments

Comments need to start with `/*` and end with `*/`. Otherwise, you can define one-line comments with `//` if you like. Comments are colored green if Java understands them correctly. Notice that Java marks the start of the unterminated comment.



```
/* Elizabeth White
A Hello World program example for class */
import java.io.*;
/**
 * This class can take a variable number of parameters on the command
 * line. Program execution begins with the main() method. The class
 * constructor is not invoked unless an object of type 'Class1'
 * created in the main() method.
 */
public class HelloWorld
{
    /**
     * The main entry point for the application.
     *
     * @param args Array of parameters passed to the application
     * via the command line.
     */
    public static void main (String[] args) throws IOException
    {
        BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
        int birthyear, age;
        String name;
        System.out.println("Hello World!");
        System.out.print("Enter your name: ");
        name = stdin.readLine();
        System.out.print("Enter your year of birth: ");
        birthyear = Integer.parseInt(stdin.readLine());
        age = 2003 - birthyear;
        System.out.println("Hello " + name + "!");
        System.out.println("This year, you will be " + age + " years old. Good luck!");
        System.out.print("Hit the Enter key to finish");
        stdin.readLine();
    }
}
```

Task List - 2 tasks			
	Description	File	Line
	Click here to add a new task		
!	Unterminated comment	D:\TA 1200 Spring\...\HelloWorld\HelloWorld.java	12
!	Expected '}'	D:\TA 1200 Spring\...\HelloWorld\HelloWorld.java	35

Missing brackets

Brackets like these { } define blocks of code. Mixing up these brackets makes Java confused about where these blocks of code begin and/or end. Notice that the error message appears at the end of the code, not where the missing bracket should be. Nesting is ok, but the counts for { and } must add up to the same number! The whole project (public class HelloWorld) is defined as one block of code (the stuff inside the two outer brackets). The main routine (the code that runs) is defined as another block, nested inside the first project code.

To find problems with the brackets, format your code with indentation—it makes catching these sneaky bugs a lot easier! Come up with a consistent, readable scheme and use it every time you write code.

```
import java.io.*;
/**
 * This class can take a variable number of parameters on the command
 * line. Program execution begins with the main() method. The class
 * constructor is not invoked unless an object of type 'Class1'
 * created in the main() method.
 */
public class HelloWorld
{
    /**
     * The main entry point for the application.
     *
     * @param args Array of parameters passed to the application
     * via the command line.
     */
    public static void main (String[] args) throws IOException
    {
        BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
        int birthyear, age;
        String name;
        System.out.println("Hello World!");
        System.out.print("Enter your name: ");
        name = stdin.readLine();
        System.out.print("Enter your year of birth: ");
        birthyear = Integer.parseInt(stdin.readLine());
        age = 2003 - birthyear;
        System.out.println("Hello " + name + "!");
        System.out.println("This year, you will be " + age + " years old. Good luck!");
        System.out.print("Hit the Enter key to finish");
        stdin.readLine();
    }
}
```

I	Description	File	Line
<input checked="" type="checkbox"/>	Click here to add a new task		
<input checked="" type="checkbox"/>	Expected }	D:\TA 1200 Spring\...\HelloWorld\HelloWorld.java	35
<input type="checkbox"/>			
<input type="checkbox"/>			
<input type="checkbox"/>			