```java
import java.util.*;

public class firstProgram {

    Run | Debug
    public static void main(String[] args) {

        // declare variables and scanner
        Scanner keyboard = Scanner(System.in);
        String name = "";

        // ask user for their name
        System.out.print(s: "What is your name: ");
        name = keyboard.nextLine();

        // print out your final statement
        System.out.println("Hey " +name+ "! Welcome to programming!");
    }
}
```

HOW TO WRITE YOUR FIRST PROGRAM

and why it's not as scary as you think

BY ALEXIS GIOBBI

The world of programming can feel like a mystical field filled with people hunched behind laptops spewing technical jargon. But really, programming is a complex field with lots of different people working in lots of different areas. There's a place for everyone in coding, regardless of gender, race, class, or age. Knowing the basics of a programming language looks great on a resume in any field of study or job sector. It can also be creative, innovative, and yes, even fun. So don't let all the stigma surrounding coding be scary, and let's write our very first program.

1. Getting everything downloaded

This is arguably the hardest step, so stick with me. Before you can start writing your program, you need to download your langauge and your IDE. Today, I'll be demonstrating everything in the Java programming language, but you pick something else that you've been wanting to learn.

To download Java, you can go to www.java.com/download and pick the right installer for your operating system (Windows or Mac.) There's tons of tutorials online if you get stuck and need some help.

The next step is downloading a text editor or an IDE (integrated development enviornment). This is where you'll write the actual code. It makes it easier to read and debug. My personal favorite is Visual Studio Code, because it's user friendly while still having wide range of capabilities. You can google that and download it too. Then you're ready to get going!

2. Save your File

This might seem self-explanatory, but it's a little different than how you would normally save something and it's important to get it right, or your program won't run. Go to the top right corner and make a new Java file. Then go to save as and name your program. It *has* to all one word.

To keep things all one word, programmers will use underscores like this `using_underscores`, or use `camelCase`, where the first word is lowercase, and the start of each of the following words is uppercase

You also want to make sure the file type is "java." If it's not automatically registering as a java file, you can physically type in fileName.java and make the file type "no extension."

3. Writing your very first line

Finally, it's time to start writing your code. Not every word is going to make sense, but that's normal! As you get more experienced and learn more about the language, everything starts to piece together. Your very first line will be:

```java
public class firstProgram {
```

A class is the basic building block of any Java program, almost like a template. The part in green, firstProgram, is what I named my program in the previous step. You ***must*** match the name here to the name of your file exactly, or your code will not be able to run. The curly bracket on the end is how Java opens and closes statements. Every single curly brace needs to be a part of a pair: one that opens, and one that closes. So go ahead and add a second brace, a closing one "}." Everything you write after this has to be between these brackets.

As you start to write more lines , the issue of spacing can be confusing. As long as you seperate each word by at least one space, you're good to go! The computer will read it the same, no matter what. I'll show you the most typical code layout, but feel free to see what works best for you.

4. Writing the main method

Between your two curly braces, you're going to write the line:

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

This is your main method. Every single program you write will need this. More complex programs have lots of methods, but you can do some really cool stuff with just the main method. All the words might look really scary, but like I said earlier, it's not important that you know exactly what each word means. Confession time: ***I don't even know what each of those words mean.*** So just re-typing the line to get it into your brain is more than fine at this point.

Again, make sure to open and close a set of curly brackets. This is the last time you'll have to do those. Even experienced programmers have trouble lining up their braces, and can spend hours trying to fix a bug, only to find their missing a bracket. So take a minute to double check yours! You should have four in total right now, and there should be nothing in between the last two but space. We'll do a code check soon, but it's good to get in the practice of looking over your code for potential problems before you even finish what you're writing.

5. Print statements

We're in the home stretch! Print statements are lines that your users can acutally see when they run the program. The majority of the code you write will never actually be seen by the user, but print statements tell the computer to literally print out a certain line so that everyone can see it. The way you write a print statement is:

```java
System.out.print(s: "Statement");
```

Don't copy the "s:", that's something Visual Studio code will put in for you automatically to show you that you're writing a print statement. Where I have the word "statement," you can put anything you want! It's really common to print "Hello World!" when you're first grasping coding, so feel free to use that if you can't think of anything else.

That semi-colon you see at the end of the line is a trademark of Java and many other programming languages. It's similar to a period at the end of a sentence, signaling that you're done with that line. Essentially every line you write will end with a bracket, if it needs to close other lines, or a semi-colon, if the line is stand-alone.

CODE CHECK

It's almost time to run what you've been working on! Before you do that, it might be helpful to look over your code and check it against mine. I only gave you a line at a time so you could get a feel of writing code on your own. Now if you're having problems or getting an error, check back with the picture below and see what's different.

```java
public class firstProgram {

    Run | Debug
    public static void main(String[] args) {

        System.out.print(s: "Hello World!");

    }
}
```

6. Running the program

The moment we've been waiting for! There's two different ways you can run your program: from the command line, or from the built-in terminal in your IDE. Java is a compiled language, meaning that the there's two steps to running the program, instead of just one. The built-in terminal will do both steps for you, and save you from a lot of confusing jargon, so that's what I'll show you here.

My instructions are for Visual Studio Code specifically, but most IDEs will have a similar process. Locate the taskbar across the top of the page and click "Run." Then click "run without debugging." If you have a debugger installed, you can also click "start debugging." A terminal should pop at the bottom of the screen and it will display some really long names, and at the very bottom it should print the line from your print statement.

If it didn't work and you got an error message, try searching for the answer on the Internet. It's hard to predict what could be wrong, and error messages are often convulted, so your best bet is consulting Google. If you didn't get an error message, but your line didn't print out, check out your print statement again and make sure it matches mine.

In addition to just utilizing Google and YouTube, programmers often will ask, answer, and search questions on a site called Stack Overflow. It's a great place to start when you're confused, but some of the answers can be overly complicated, so don't feel overwhelmed if it's confusing.

Solving problems in your code is called debugging. It can be a long, frutstrating process, even with a short, simple program. So if it's something's wrong, try not to get discouraged. Practice will help you find issues faster. If you have a friend learning with you, or a mentor, ask them to look over your code. Sometimes a fresh set of eyes is all you need to spot the mistake.

7. Celebrate!

You did it! Your very first program is complete. I hope it wasn't too confusing and you plan to code again soon. If you want to play around right now, try modifying or adding more print statements.

If you're not in the mood to do more, don't push it. Take a break, come back later. It is so awesome that you took this first step, and I hope you're now equipped with the right resources to learn it on your own. Keep coding and rock on!

```
System.out.println(x: "You can do this.");
```