Lesson 1-1: Our First Java Programs

Computer Science 46A: Introduction to Programming

San José State University

Announcements

- Change in Syllabus.
  - We will be using Zybooks.
- Lab #1 is this Friday 1/31

Learning Objectives

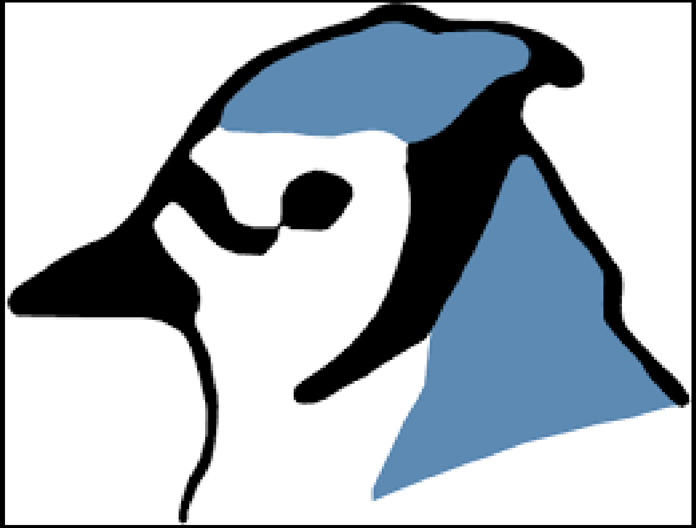
By the end of this lesson, you should be able to:

1. Write a program to print messages to the terminal
2. Identify and fix syntax errors associated with print commands
3. Use good style guidelines to keep code clean

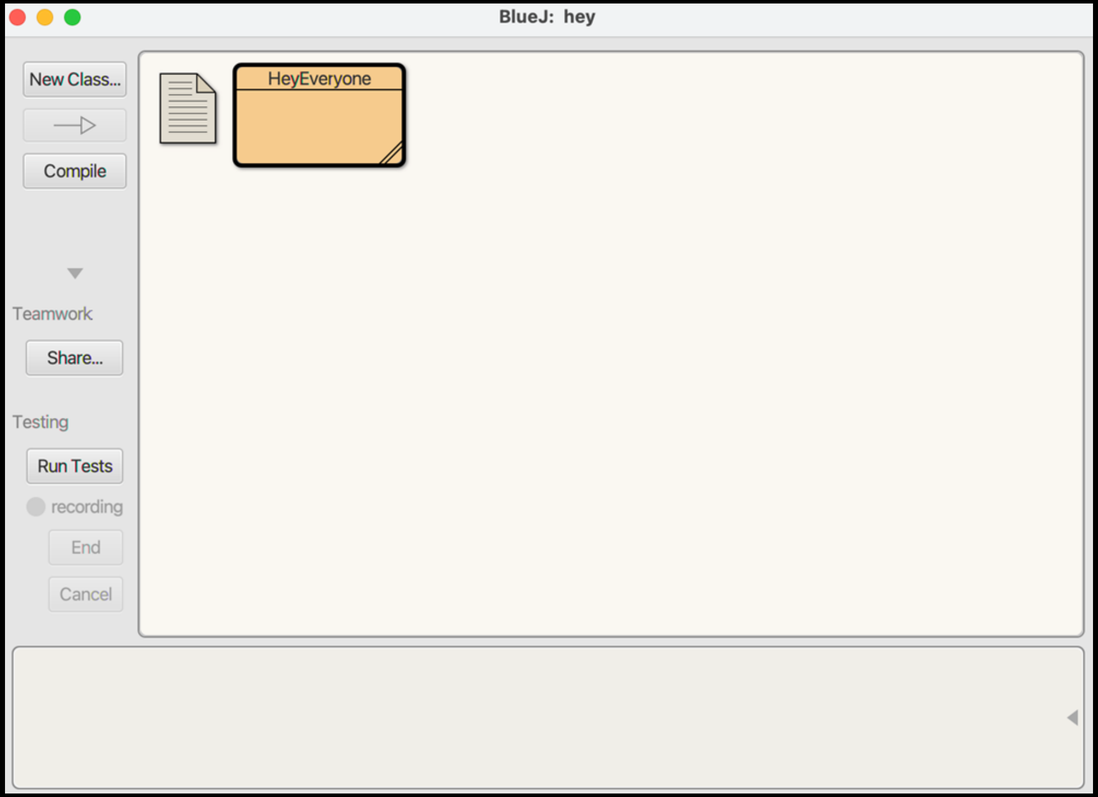
Review: Compiling and Running a Java Program

IDE: Integrated Development Environment

- An IDE is a convenient place to write, compile, and run code
- There are many different types of IDEs each with pros/cons
  - You might find that you like different IDEs for different programming languages
- In this course, we will use the BlueJ IDE
  - You should have downloaded and familiarized yourself with BlueJ
  - To demonstrate its use, we used a simple program that prints a message to the screen

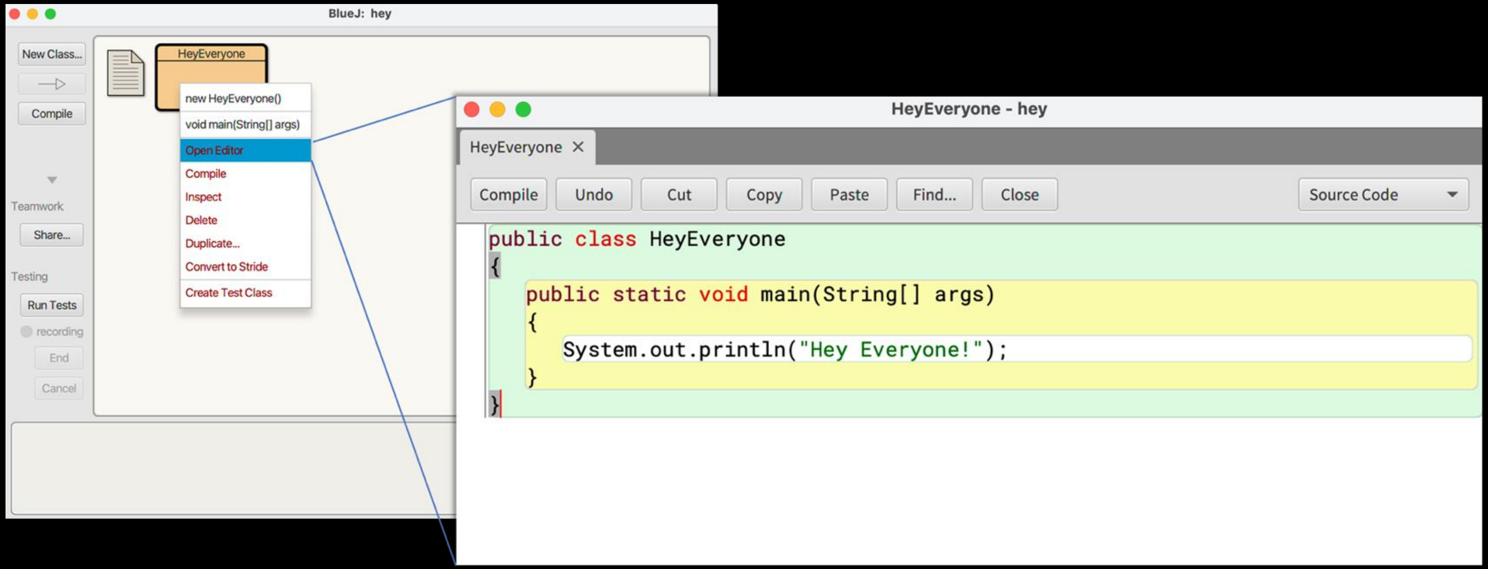


Using BlueJ: Viewing Classes



- A project called “hey” with a class called “HeyEveryone” as shown in the BlueJ editor

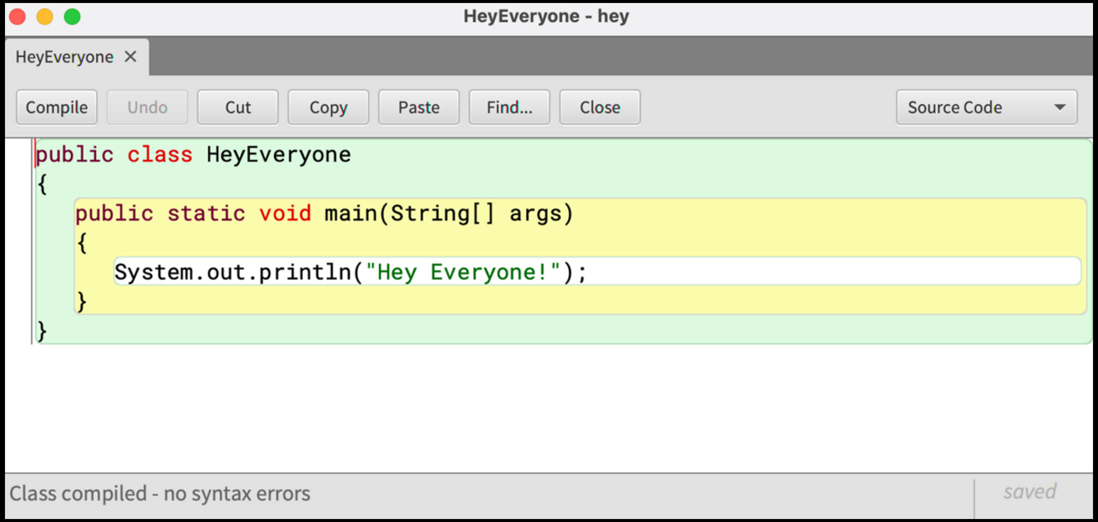
Using BlueJ: Opening the Editor



```
public class HeyEveryone
{
    public static void main(String[] args)
    {
        System.out.println("Hey Everyone!");
    }
}
```

- To open the editor for a class, right-click on the class box and choose Open Editor
- In the editor window, you can edit your code
- Notice how the editor changes the color of different words and text blocks – this is the magic of an IDE!

Using BlueJ: Using the Editor



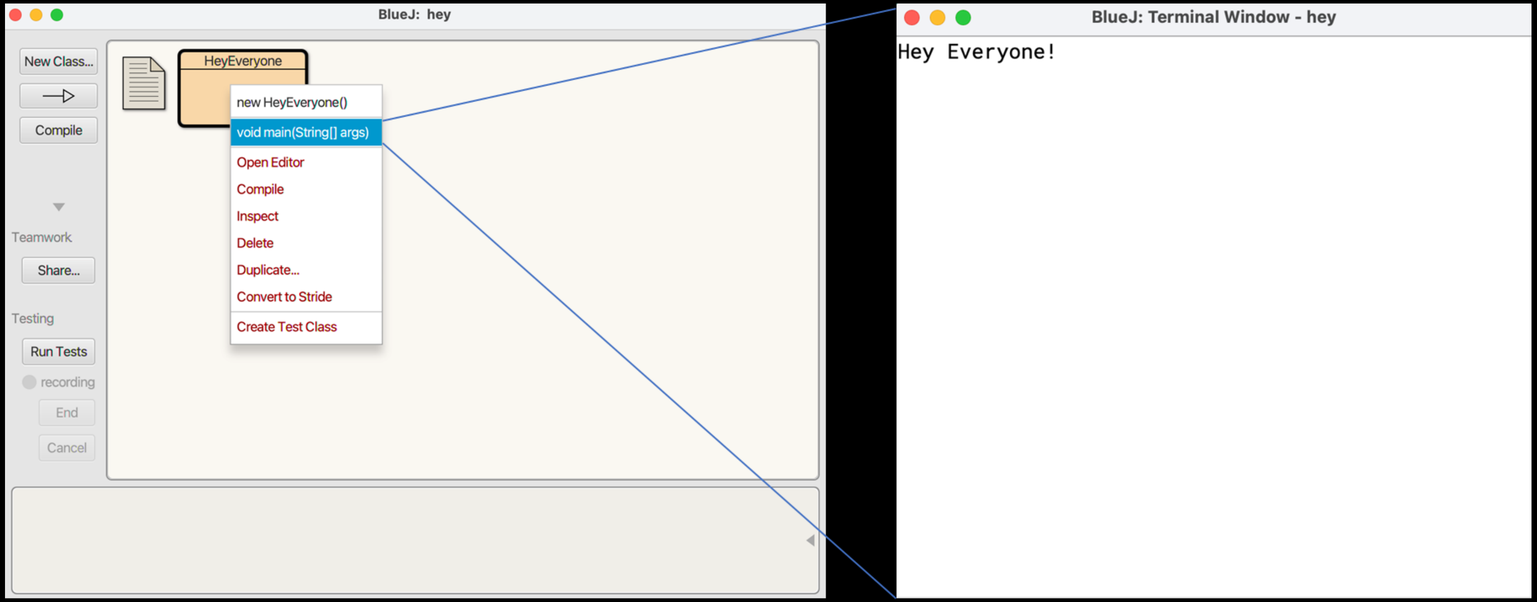
```
public class HeyEveryone
{
    public static void main(String[] args)
    {
        System.out.println("Hey Everyone!");
    }
}
```

Class compiled - no syntax errors

saved

- To compile the program, use the compile button
- When compiled successfully, you get a nice note in the bottom of the screen

Using BlueJ



```
Hey Everyone!
```

- To run the program, right click on the class and choose void main(String[] args)
- This program will print Hey Everyone! To the terminal

Printing Output to the Terminal

Note: In programming, “printing” means to display text on the screen - it has nothing to do with a printer, paper, or ink!

Breaking down HeyEveryone.java

```java
public class HeyEveryone
{
    public static void main(String[] args)
    {
        System.out.println("Hey Everyone!");
    }
}
```

Breaking down HeyEveryone.java: Declaring the Class

(1) Declare the class $\left\{ \right.$

```
public class HeyEveryone
{
    public static void main(String[] args)
    {
        System.out.println("Hey Everyone!");
    }
}
```

Breaking down HeyEveryone.java: The Class Name

(2) Class Name

```
public class HeyEveryone
{
    public static void main(String[] args)
    {
        System.out.println("Hey Everyone!");
    }
}
```

Breaking down HeyEveryone.java: Declaring a Method

(3) Declare a method in the class

```
public class HeyEveryone
{
    public static void main(String[] args)
    {
        System.out.println("Hey Everyone!");
    }
}
```

Breaking down HeyEveryone.java: Method Arguments

(4) This action will deal with strings

```
public class HeyEveryone
{
    public static void main(String[] args)
    {
        System.out.println("Hey Everyone!");
    }
}
```

Breaking down HeyEveryone.java: Calling a Method

```
public class HeyEveryone
{
    public static void main(String[] args)
    {
        System.out.println("Hey Everyone!");
    }
}
```

(5) Call a method to print output to the screen

Note: adding ln at the end adds a new line

Breaking down `HeyEveryone.java`: The String Output

```
public class HeyEveryone
{
    public static void main(String[] args)
    {
        System.out.println("Hey Everyone!");
    }
}
```

(6) The string to output

Breaking down `HeyEveryone.java` : A few notes

The class name matches the file name

```
public class HeyEveryone
{
   public static void main(String[] args)
   {
      System.out.println("Hey Everyone!");
   }
}
```

Every open bracket has a closed backet partner

Words are case sensitive. E.g. System is capitalized while print is lowercase

Lines with statements always end in a semi-colon

Breaking down HeyEveryone.java: A few more notes

In the start of our course, these components won't change (except for the class name)

```
public class HeyEveryone
{
    public static void main(String[] args)
    {
        System.out.println("Hey Everyone!");
    }
}
```

This section is where we will make all of our edits

Navigate to the Canvas course space and download the Lecture01-1.zip file which contains today's Participation Examples

Syntax

Syntax Overview

- Rules for how pieces of code are arranged
- If you violate one of the rules, you will get an error when you try to compile your program
- BlueJ is a smart IDE that will warn you if you are committing a syntax error before you compile.

Common Syntax Errors

```
System.out.println("Hey Everyone!");
```

1. Incorrect capitalization (`System` should be capitalized, `out` should not)
2. Incorrect method

   (`print` or `println` is used for printing, not any other word)

1. Lines must end in a semi-colon (`;`)
2. Strings must be wrapped in double quotation marks (“ “)

Participation Exercise 1-1a: `FixingSyntaxError` Instructions

Steps to get started with Participation Exercises:

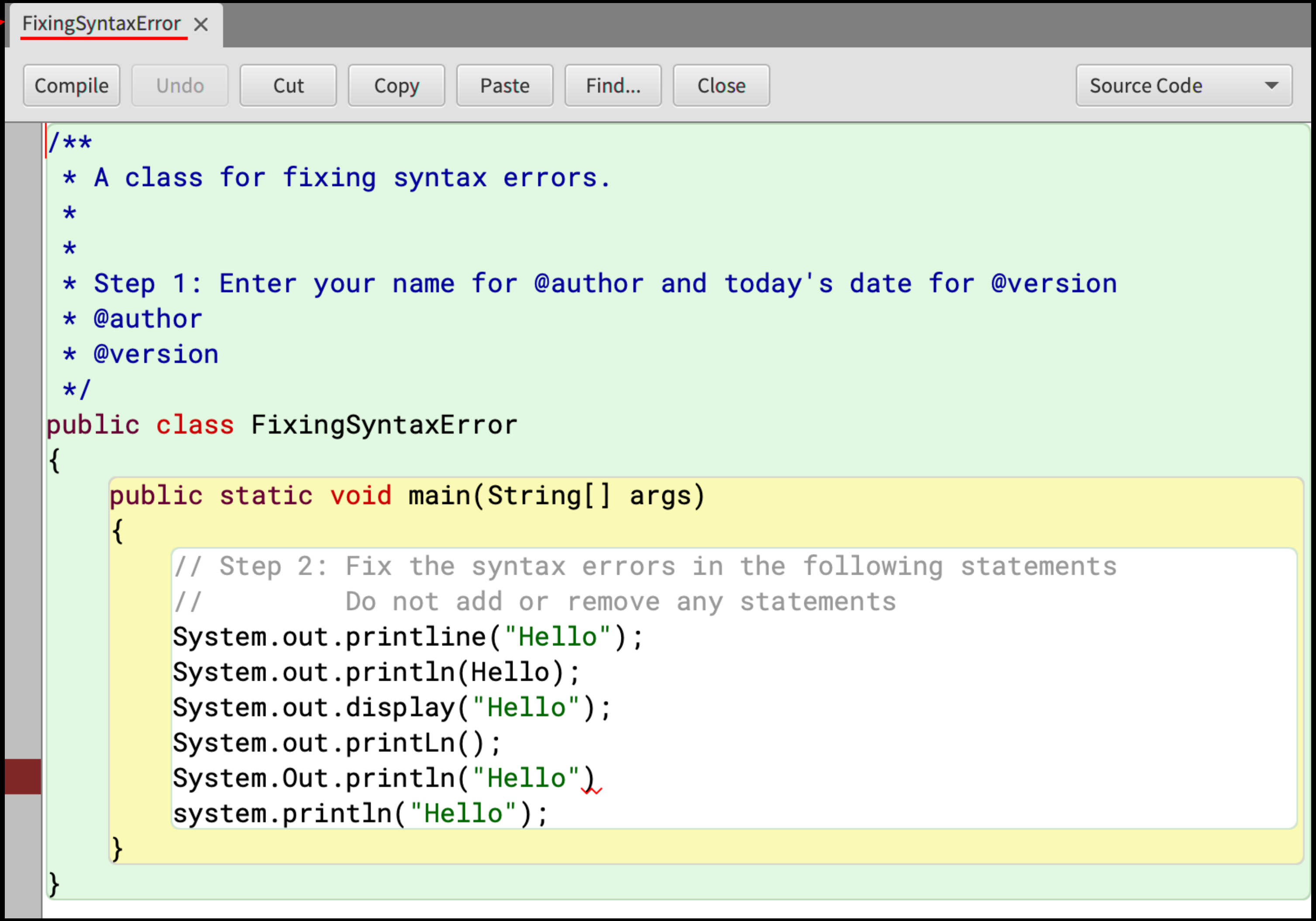
1. Be sure you have downloaded Lesson01-1.zip from Canvas
2. Unzip and export the Lesson01-1.zip file to access its contents
3. Copy the par01-1a folder into your par01-1 directory
4. Double click on the package.bluej file to open the exercise in BlueJ
5. Begin Editing

BlueJ: FixingSyntaxError.java Errors

Red underline indicates this file has a syntax error

The BlueJ IDE will identify some (but not all) syntax errors before you even run the program

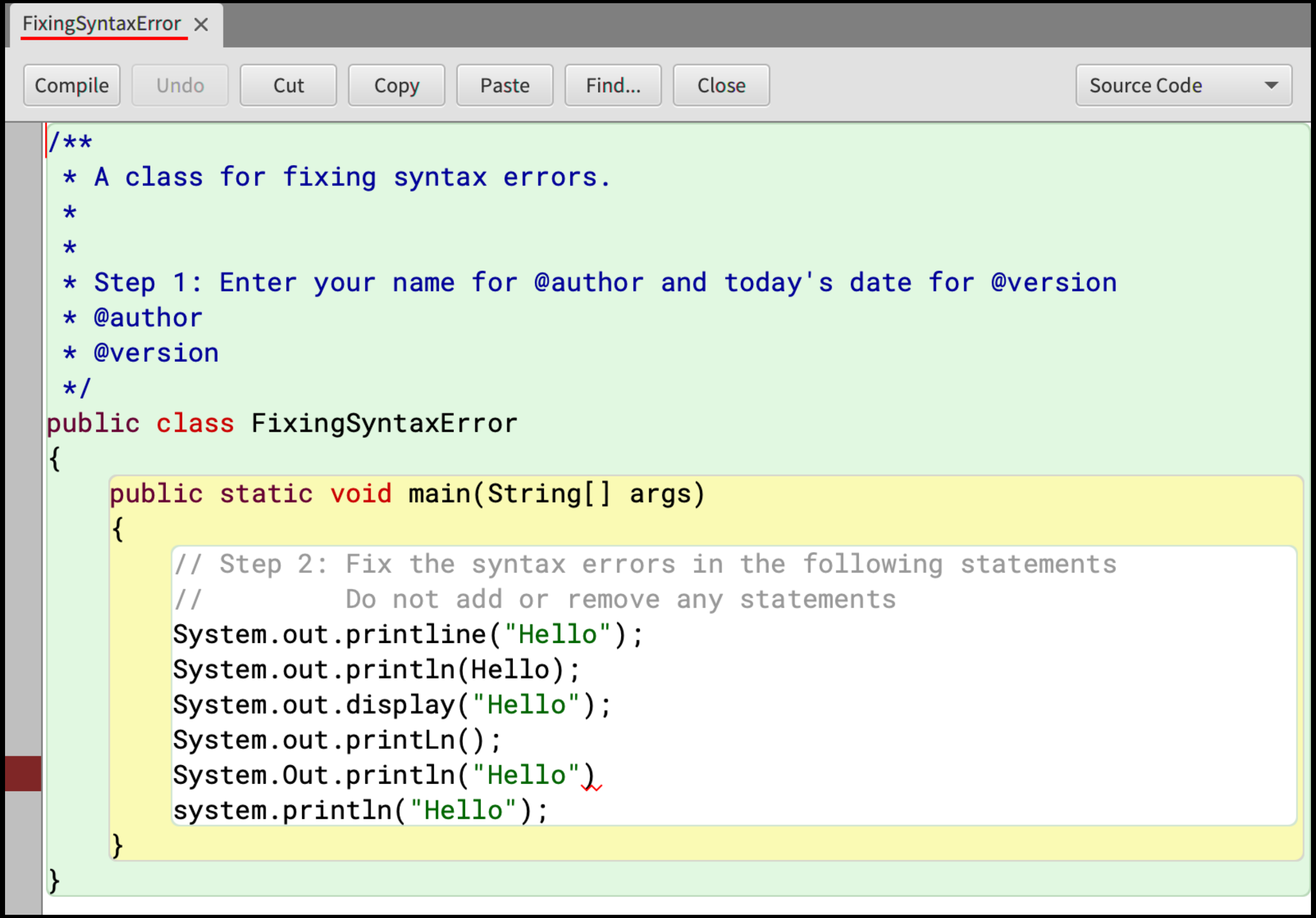
Red block on a line indicates this line has a syntax error



```
/**
 * A class for fixing syntax errors.
 *
 *
 * Step 1: Enter your name for @author and today's date for @version
 * @author
 * @version
 */
public class FixingSyntaxError
{
    public static void main(String[] args)
    {
        // Step 2: Fix the syntax errors in the following statements
        //         Do not add or remove any statements
        System.out.printline("Hello");
        System.out.println(Hello);
        System.out.display("Hello");
        System.out.printLn();
        System.Out.println("Hello")
        system.println("Hello");
    }
}
```

The class FixingSyntaxError in BlueJ

BlueJ: FixingSyntaxError.java Comments



```java
/**
 * A class for fixing syntax errors.
 *
 *
 * Step 1: Enter your name for @author and today's date for @version
 * @author
 * @version
 */
public class FixingSyntaxError
{
    public static void main(String[] args)
    {
        // Step 2: Fix the syntax errors in the following statements
        //         Do not add or remove any statements
        System.out.printline("Hello");
        System.out.println(Hello);
        System.out.display("Hello");
        System.out.printLn();
        System.Out.println("Hello")
        system.println("Hello");
    }
}
```

Comments can be in multiple lines bound between /* and */ characters

Or comments can be on a single line that starts with two // characters

Comments are ignored by the compiler

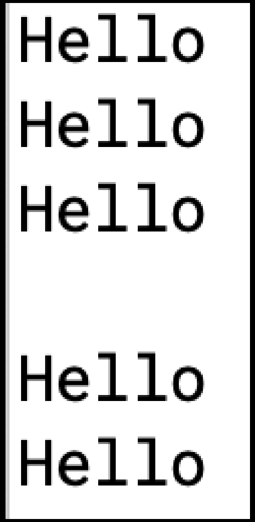
You can add as many as you like with no impact on the code

It is a very good habit to add comments to your code so that you (and others) can understand what each piece is intended to do

Participation Exercise 1-1a: `FixingSyntaxError`

Goal: Edit a program that has several different syntax errors.

When all errors have been fixed, the program will compile without error and generate the output at the right.



```
Hello
Hello
Hello

Hello
Hello
```

Output of `FixingSyntaxError`

Codecheck Link: HERE and on Canvas

CodeCheck

- Introduced in homework 0
- Tool to check the output of code
- Will be used in this course to
  - Check whether your code works as expected
  - Provide code summaries to use for submission on Canvas
- I will add the CodeCheck links for all homework and participation exercises on Canvas each week
- For this example, we will use the CodeCheck link HERE

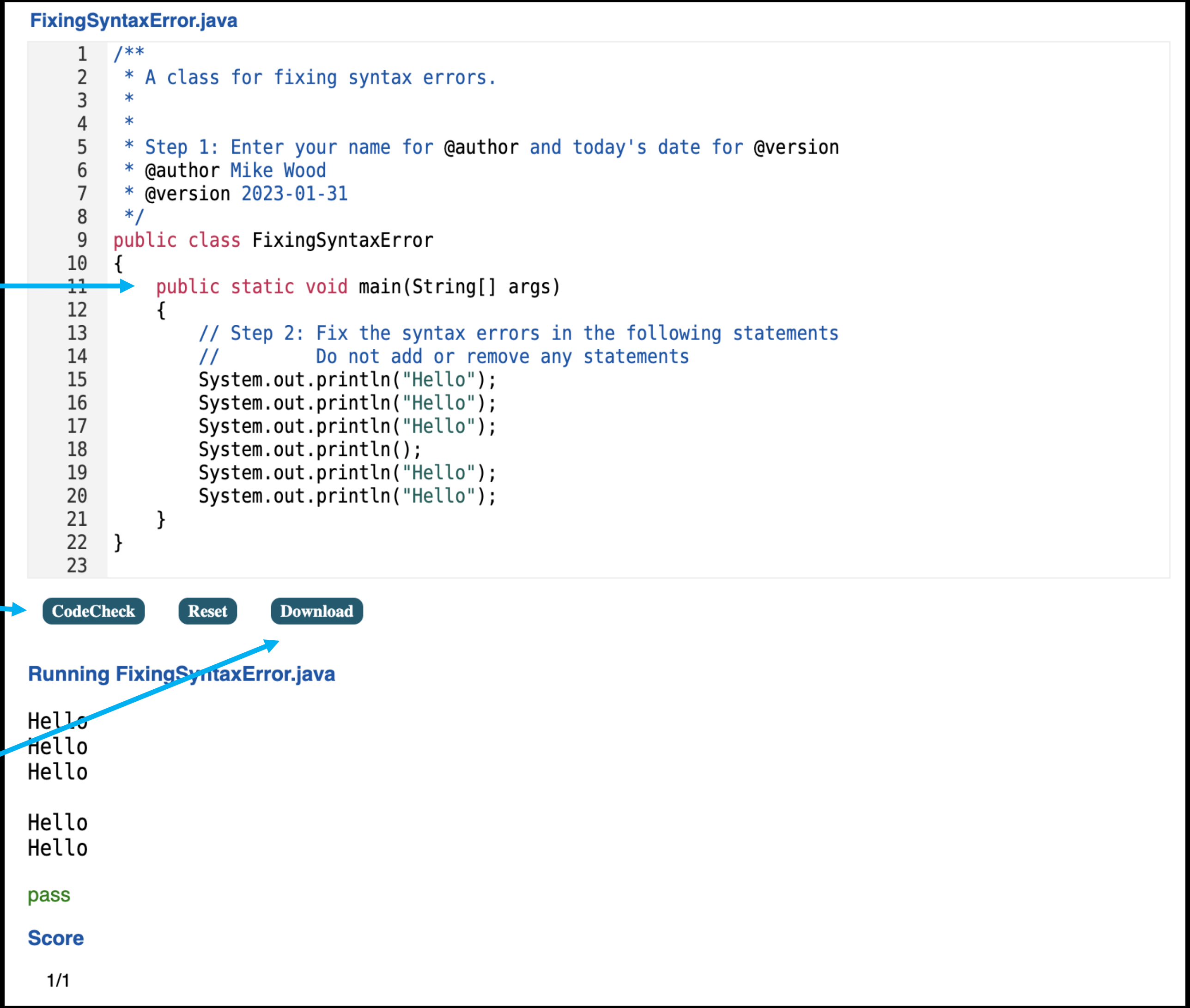
Check your work in CodeCheck: `FixingSyntaxError`

(1) Be sure the file name matches the file you are submitting

(2) Copy and paste your code in the box, overwriting any existing code

(3) Press the CodeCheck button to verify your work and create your report

(4) Download the report when you receive a passing score



FixingSyntaxError.java

```java
/**
 * A class for fixing syntax errors.
 *
 *
 * Step 1: Enter your name for @author and today's date for @version
 * @author Mike Wood
 * @version 2023-01-31
 */
public class FixingSyntaxError
{
    public static void main(String[] args)
    {
        // Step 2: Fix the syntax errors in the following statements
        //         Do not add or remove any statements
        System.out.println("Hello");
        System.out.println("Hello");
        System.out.println("Hello");
        System.out.println();
        System.out.println("Hello");
        System.out.println("Hello");
    }
}
```

Running FixingSyntaxError.java

```
Hello
Hello
Hello

Hello
Hello
```

pass

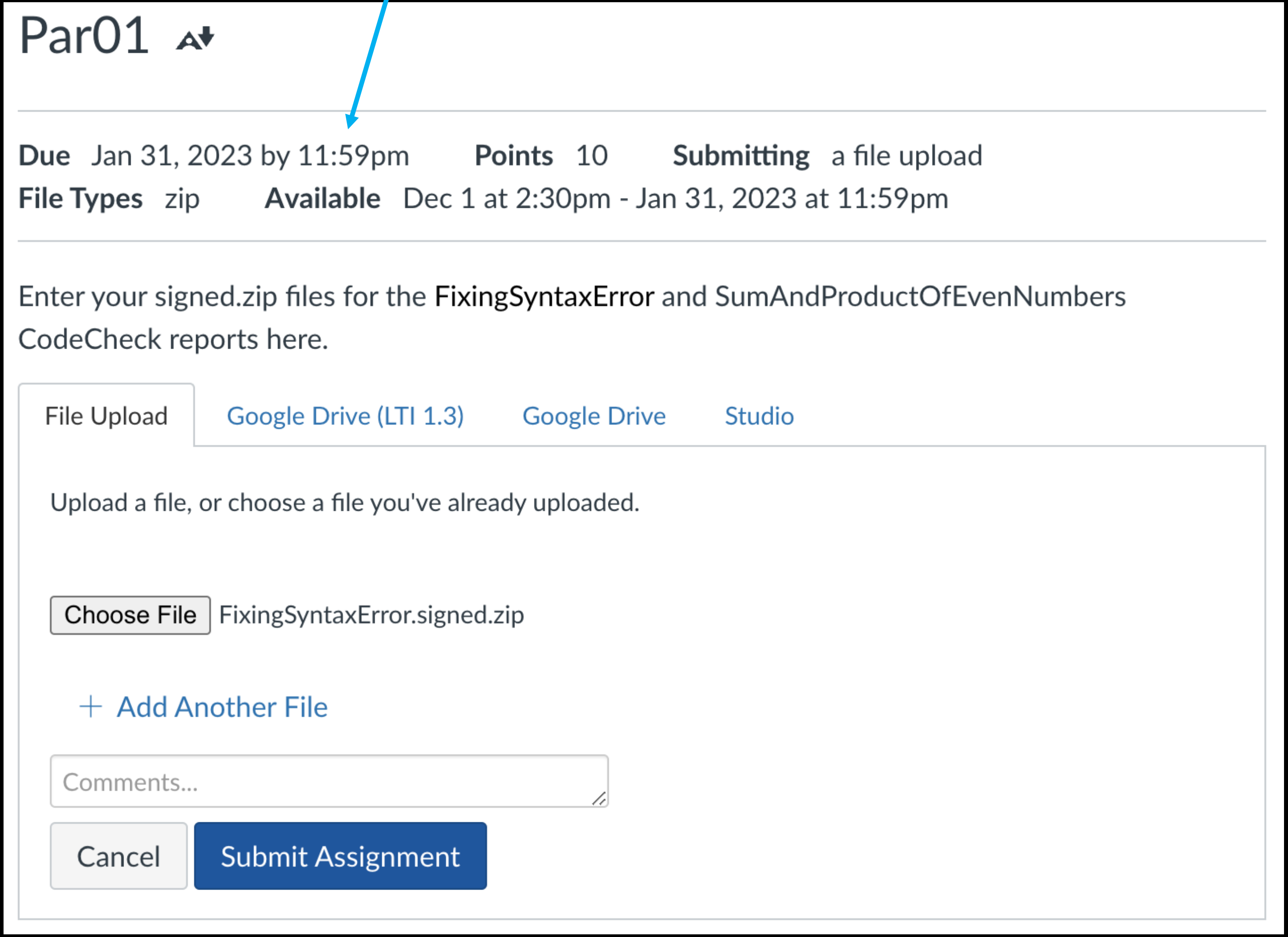
Score

1/1

Submit your work on Canvas: FixingSyntaxError

Note! The par examples are due by the end of the following day for each lecture

Upload the file to par01-1



Par01

Due Jan 31, 2023 by 11:59pm **Points** 10 **Submitting** a file upload
File Types zip **Available** Dec 1 at 2:30pm - Jan 31, 2023 at 11:59pm

Enter your signed.zip files for the FixingSyntaxError and SumAndProductOfEvenNumbers CodeCheck reports here.

Upload a file, or choose a file you've already uploaded.

FixingSyntaxError.signed.zip

Don't click "Submit Assignment" just yet – we have one more participation example left to do

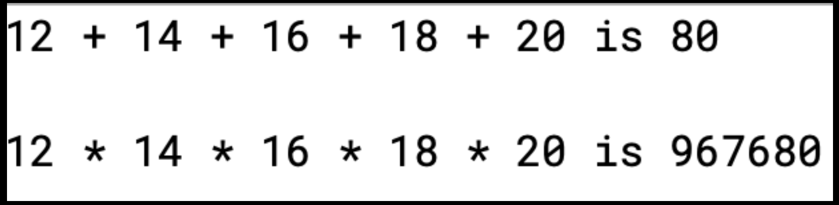
Summary of Typical Steps for Assignments

1. Access Canvas for the files for the exercise
2. Start a project in BlueJ
3. Edit, compile, and run code in BlueJ
4. When complete, move code to CodeCheck for verification
5. Download zip file and submit to Canvas for credit

Participation Exercise 1-1b: `SumAndProductOfEvenNumbers`

Goal: Write a program to print the sum and product of even numbers

E.g. Print the sum and product of all even numbers between 12 and 20 (inclusive).

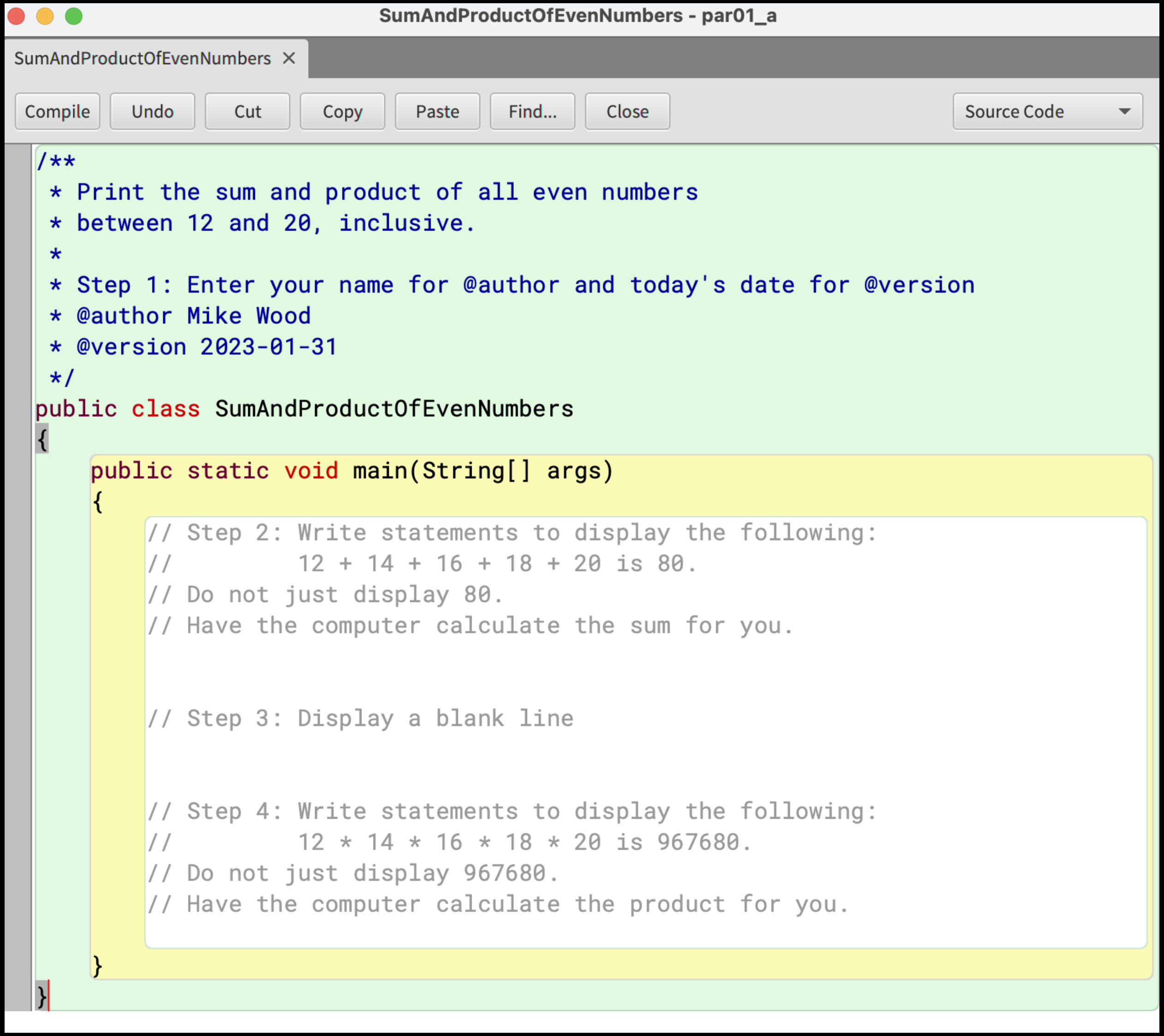


```
12 + 14 + 16 + 18 + 20 is 80

12 * 14 * 16 * 18 * 20 is 967680
```

Output of `SumAndProductOfEvenNumbers`

Codecheck Link: HERE and on Canvas

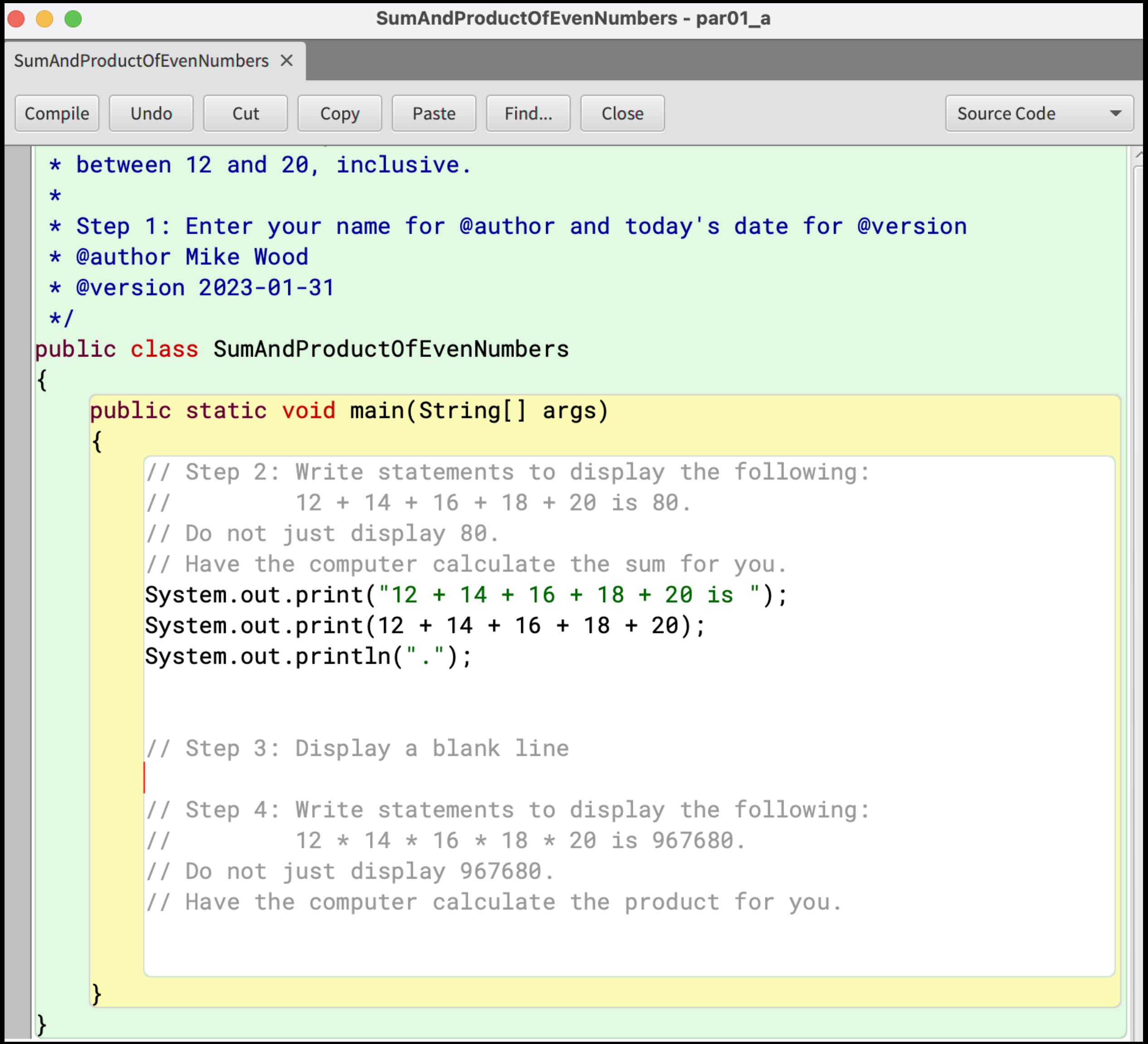
BlueJ: SumAndProductOfEvenNumbers


SumAndProductOfEvenNumbers - par01_a
SumAndProductOfEvenNumbers ✕
Compile
Undo
Cut
Copy
Paste
Find...
Close
Source Code

```java
/**
 * Print the sum and product of all even numbers
 * between 12 and 20, inclusive.
 *
 * Step 1: Enter your name for @author and today's date for @version
 * @author Mike Wood
 * @version 2023-01-31
 */
public class SumAndProductOfEvenNumbers
{
    public static void main(String[] args)
    {
        // Step 2: Write statements to display the following:
        //         12 + 14 + 16 + 18 + 20 is 80.
        // Do not just display 80.
        // Have the computer calculate the sum for you.


        // Step 3: Display a blank line


        // Step 4: Write statements to display the following:
        //         12 * 14 * 16 * 18 * 20 is 967680.
        // Do not just display 967680.
        // Have the computer calculate the product for you.
    }
}
```

Follow the comments written for Step 2 to complete the program

Try it for yourself


SumAndProductOfEvenNumbers - par01_a
SumAndProductOfEvenNumbers ×
Compile
Undo
Cut
Copy
Paste
Find...
Close
Source Code

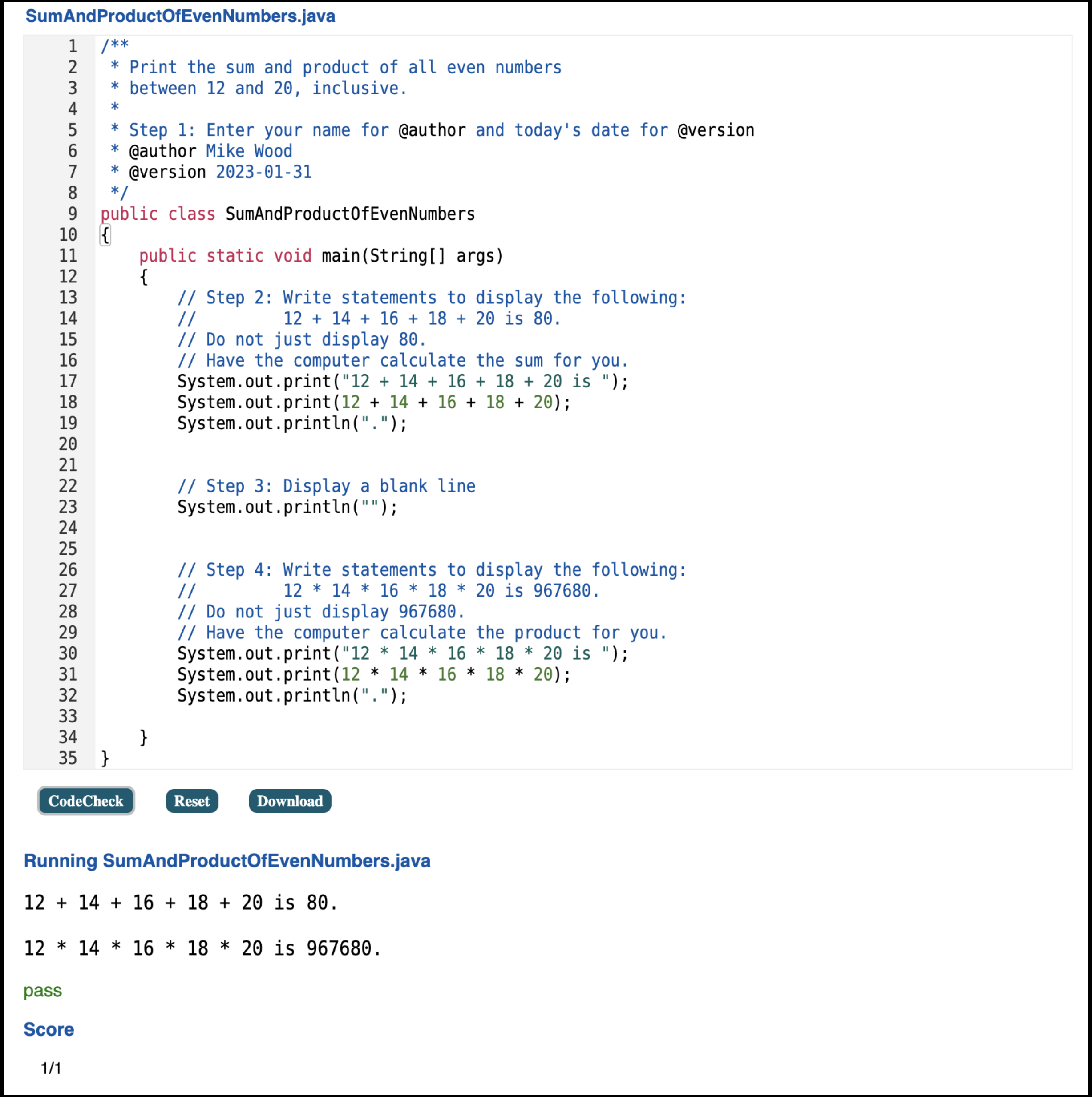
```
 * between 12 and 20, inclusive.
 *
 * Step 1: Enter your name for @author and today's date for @version
 * @author Mike Wood
 * @version 2023-01-31
 */
public class SumAndProductOfEvenNumbers
{
    public static void main(String[] args)
    {
        // Step 2: Write statements to display the following:
        //         12 + 14 + 16 + 18 + 20 is 80.
        // Do not just display 80.
        // Have the computer calculate the sum for you.
        System.out.print("12 + 14 + 16 + 18 + 20 is ");
        System.out.print(12 + 14 + 16 + 18 + 20);
        System.out.println(".");


        // Step 3: Display a blank line

        // Step 4: Write statements to display the following:
        //         12 * 14 * 16 * 18 * 20 is 967680.
        // Do not just display 967680.
        // Have the computer calculate the product for you.

    }
}
```

Follow the comments written for Steps 3 and 4 to complete the program

When you finish, use CodeCheck to check your work and create your participation file

Check your work in CodeCheck: SumAndProductOfEvenNumbers

Click download to receive a zip file which you will submit to Canvas



SumAndProductOfEvenNumbers.java

```java
/**
 * Print the sum and product of all even numbers
 * between 12 and 20, inclusive.
 *
 * Step 1: Enter your name for @author and today's date for @version
 * @author Mike Wood
 * @version 2023-01-31
 */
public class SumAndProductOfEvenNumbers
{
    public static void main(String[] args)
    {
        // Step 2: Write statements to display the following:
        //         12 + 14 + 16 + 18 + 20 is 80.
        // Do not just display 80.
        // Have the computer calculate the sum for you.
        System.out.print("12 + 14 + 16 + 18 + 20 is ");
        System.out.print(12 + 14 + 16 + 18 + 20);
        System.out.println(".");


        // Step 3: Display a blank line
        System.out.println("");


        // Step 4: Write statements to display the following:
        //         12 * 14 * 16 * 18 * 20 is 967680.
        // Do not just display 967680.
        // Have the computer calculate the product for you.
        System.out.print("12 * 14 * 16 * 18 * 20 is ");
        System.out.print(12 * 14 * 16 * 18 * 20);
        System.out.println(".");

    }
}
```

Running SumAndProductOfEvenNumbers.java

```
12 + 14 + 16 + 18 + 20 is 80.

12 * 14 * 16 * 18 * 20 is 967680.
```

pass

Score

1/1

Let’s Talk About Style

Style in Programming

- A computer does not care how “pretty” your code is – if it follows all of the rules, then the program will compile and run
- BUT a well-written code is easier to understand and debug
- This is especially important when you are sharing code with others - something you will do no matter what purpose you are coding for
- In this class, style will be a part of your grade on assignments

Style: Braces

Good Style 😃

Braces on separate lines

```
public class HeyEveryone
{
   public static void main(String[] args)
   {
      System.out.println("Hey Everyone!");
   }
}
```

Bad Style 😡

Braces are stacked and hidden at end of the lines

```
public class HeyEveryone
{
   public static void main(String[] args){
      System.out.println("Hey Everyone!");}
}
```

Style: Indentation

Use 3 or 4 spaces, your choice; If you also use Python, use 4!

Good Style

Consistent indentation for code blocks

```
public class HeyEveryone
{
   public static void main(String[] args)
   {
      System.out.println("Hey Everyone!");
   }
}
```

Bad Style

No spacing or inconsistent spacing

```
public class HeyEveryone
{
public static void main(String[] args)
{
  System.out.println("Hey Everyone!");
}
}
```

Style: Spaces

Good Style 😃

Statements, numbers, and operators have one space between them making it easy to read

```
public class SumAndProductOfEvenNumbers
{
    public static void main(String[] args)
    {
        System.out.print(12 + 14 + 16 + 18 + 20);
    }
}
```

Bad Style 😡

Spacing is inconsistent or things are mashed together

```
public class SumAndProductOfEvenNumbers
{
    public     static     void main(String[] args)
    {
        System.out.print(12+14+16+18+20);
    }
}
```

Style: Header Comment Block

Good Style 😃

Header explains briefly what the program does and has tags for the author and version

```
/**
* Print the sum and product of all even numbers
* between 12 and 20, inclusive.
*
* @author Mike Wood
* @version 2023-01-31
*/
```

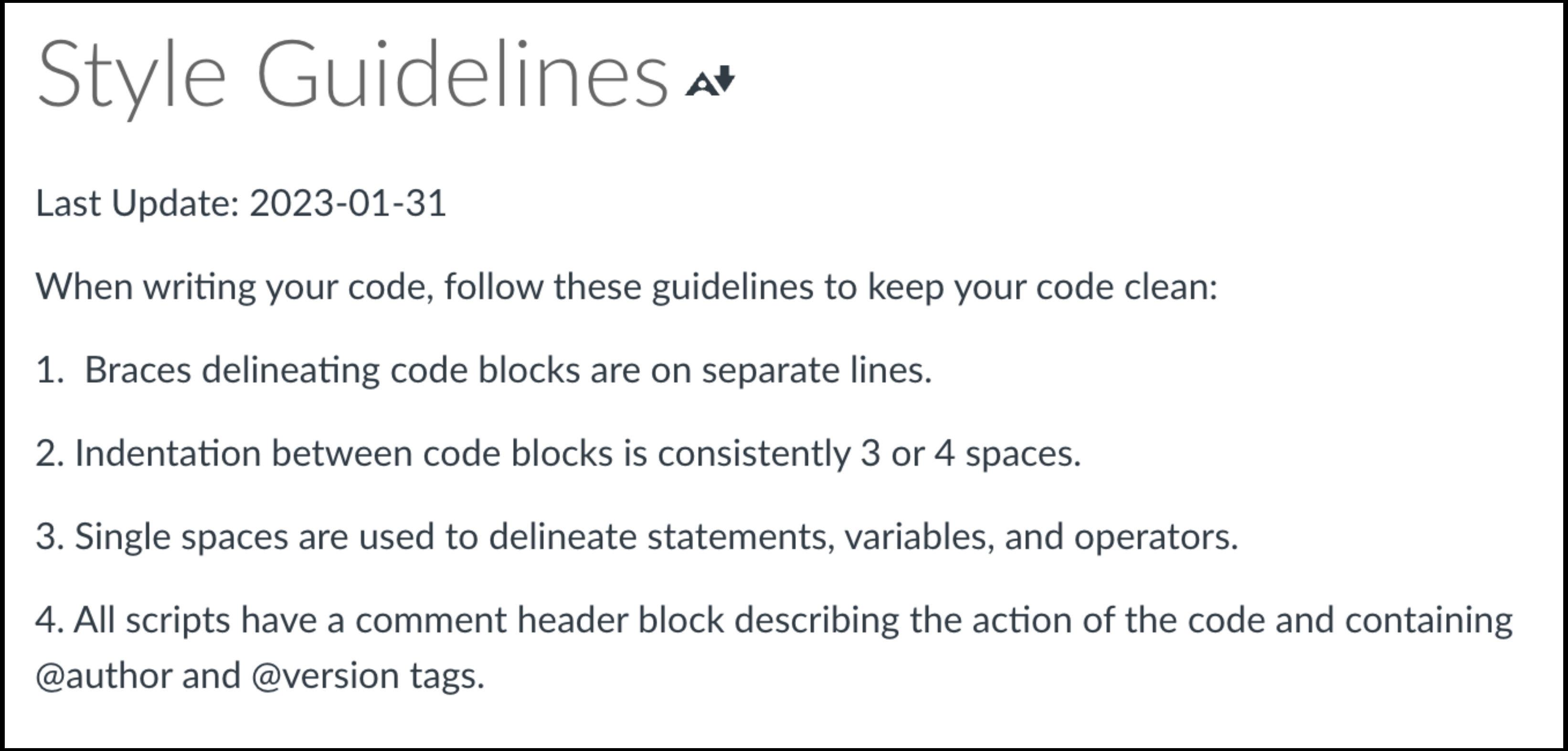
Bad Style 😡

Header is non-descriptive and/or does not have tags for the author or version

```
/**
* par01b.
*
* @author
* @version
*/
```

Style Guidelines on Canvas

You can access the style guidelines on Canvas:



Style Guidelines

Last Update: 2023-01-31

When writing your code, follow these guidelines to keep your code clean:

1. Braces delineating code blocks are on separate lines.

2. Indentation between code blocks is consistently 3 or 4 spaces.

3. Single spaces are used to delineate statements, variables, and operators.

4. All scripts have a comment header block describing the action of the code and containing @author and @version tags.

For Next Time

- Please watch announcements for Reading assignments/Homeworks
- Lab #1 is this Friday (2/2)