2020 CI401

Introduction to programming

Week 1.02
Variables, loops and choices

Dr Roger Evans

Module leader

13th October 2020

Module structure (version 1)

Semester 1

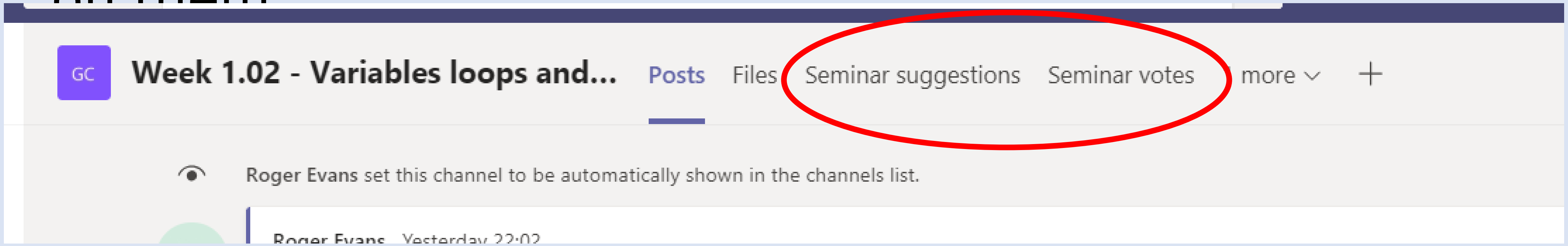
Week	Topic	Theme
1.01	Introduction / Hello World	Coding
1.02	Variables, loops and choices	Coding
1.03	Input, more loops and choices	Coding
1.04	Types and expressions	Coding
1.05	Let’s play Top Trumps!	Data
1.06	Objects and classes	OO
1.07	Getting organised	Data
1.08	Working with numbers	Data
1.09	Simple Algorithms	Dvp
1.10	Introduction to JavaFX	Dvp
1.11	Simple Animation	Dvp
	Xmas vacation 21 Dec – 8 Jan	
1.12	GUIs using MVC	OO
1.13		

Semester 2

Week	Topic	Theme	Project
2.01	Project topics and assessment	Project	Set
2.02	Simple Inheritance	OO	Lab
2.03	Scope, Visibility and Encapsulation	OO	Lab
2.04	Testing - JUnit	Testing	Lab
2.05	Documentation - Javadoc	Doc	Study
2.06	Collections and generic types	Data	Study
2.07	IO: files and streams	Dvp	Study
	Easter Vacation 29 Mar – 16 Apr		
2.08	Numbers – the computer’s view	Data	Submit?
2.09	Java vs Python		
2.10	More algorithms – search and sort	Dvp	
2.11	How fast is my code?	Dvp	
2.12	Java ‘under the hood’		
2.13	Revision week		

Seminars

- The seminars are now running on Mondays at 1200
- The introduction slides we used this week are available in Study materials for this week
- In the teams channel for this week, there are tabs containing forms to suggest seminar topics, and to vote on them



Roger Evans set this channel to be automatically shown in the channels list.

Roger Evans Yesterday 22:02

Review of week 1.01

Java application:

- 'Hello world' program
- Edit-compile-run process

Code: essential, good style, helpful

- Capital letters
- Punctuation (semicolons, brackets)
- Spaces and new lines
- Colours (syntax highlighting)

Simple instructions

```
System.out.println("Hello World");
```

Sequences

```
System.out.println("Hello World");
System.out.println("Hello UK");
System.out.println("Hello Brighton");
```

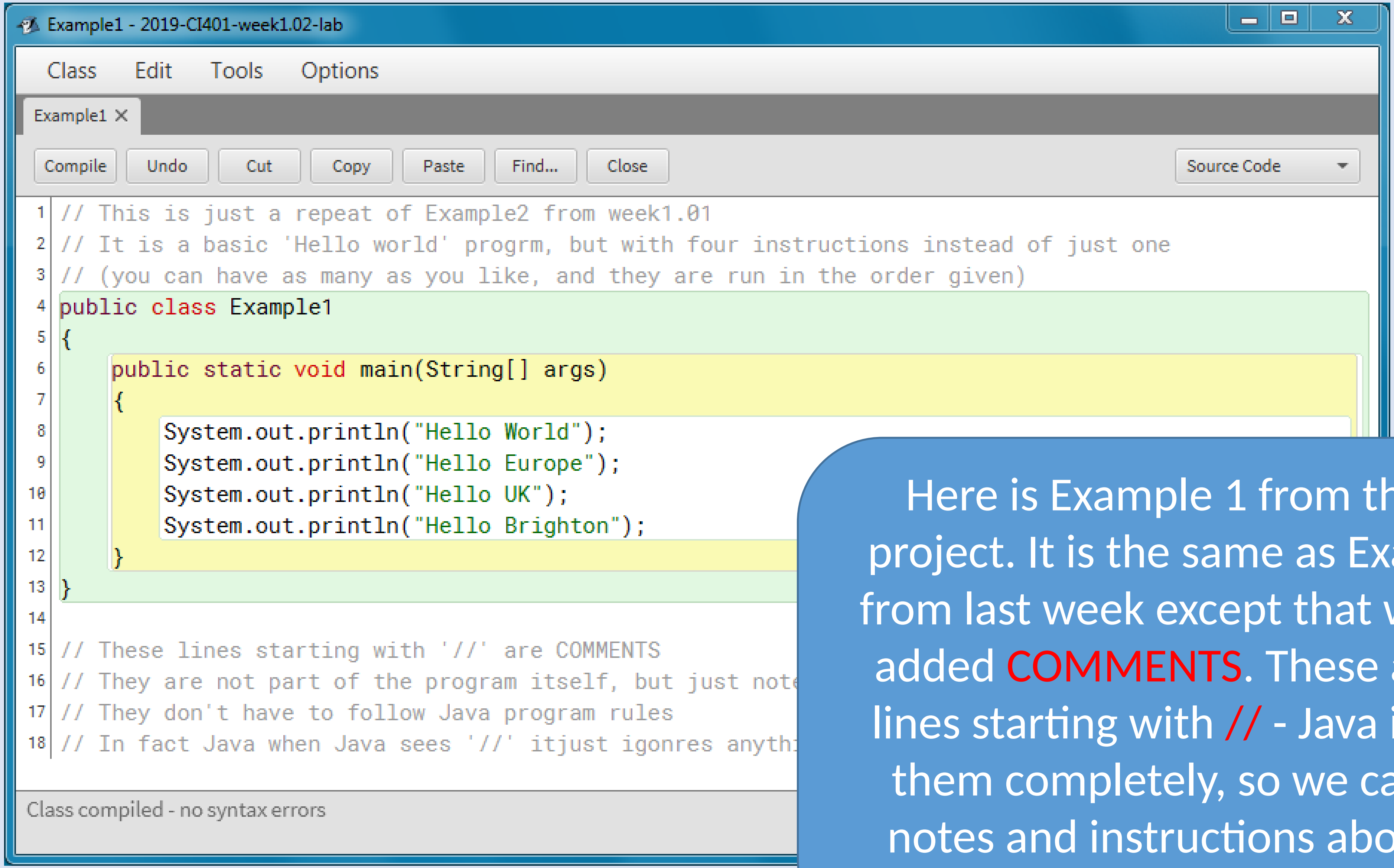
Numbers, 'doing sums'

```
25, 25+3, 3.14*2*2
```

Strings, 'adding strings together'

```
"Hello Brighton"
"Hello " + "Brighton"
```

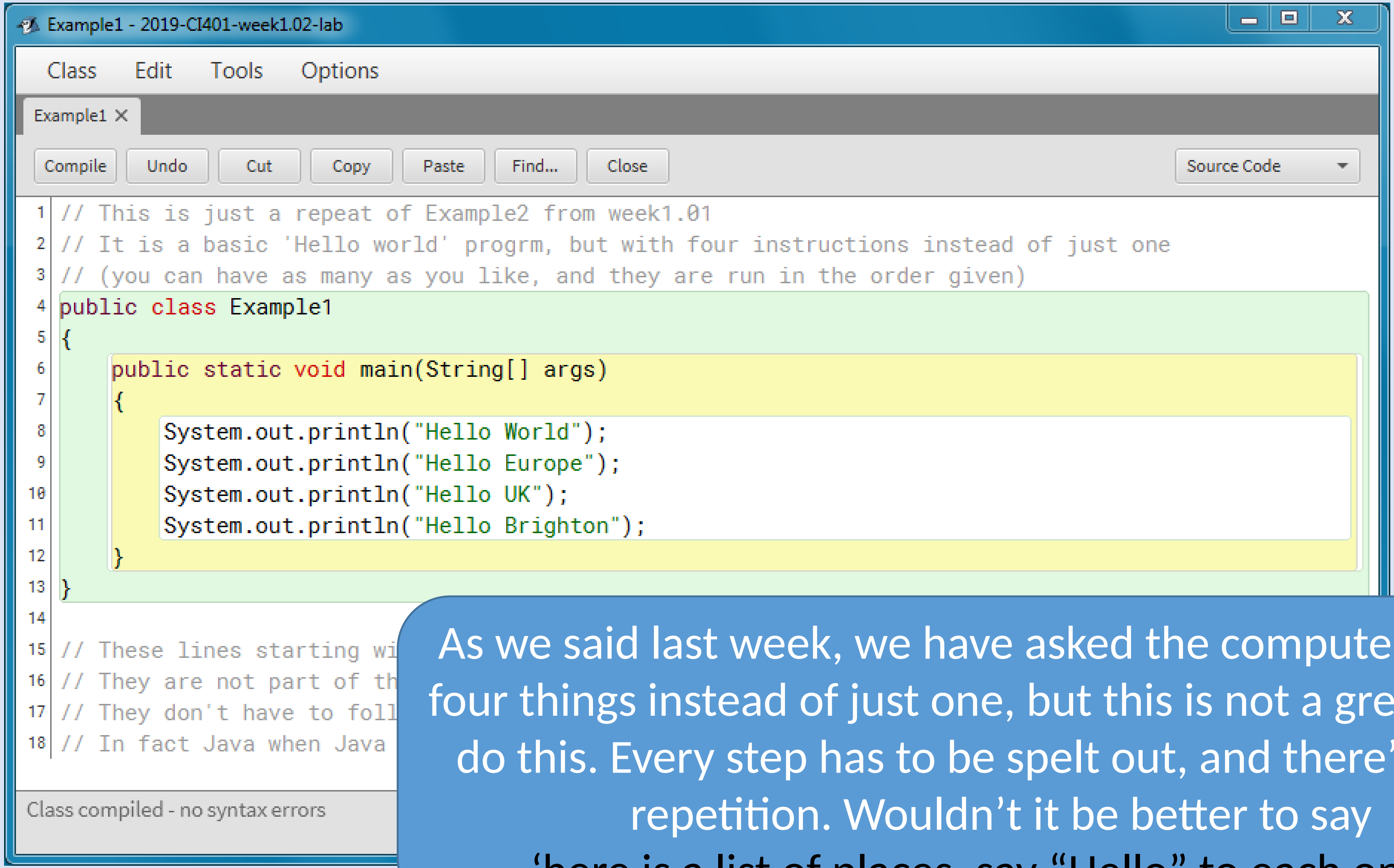
Variables and loops



```
// This is just a repeat of Example2 from week1.01
// It is a basic 'Hello world' progrm, but with four instructions instead of just one
// (you can have as many as you like, and they are run in the order given)
public class Example1
{
    public static void main(String[] args)
    {
        System.out.println("Hello World");
        System.out.println("Hello Europe");
        System.out.println("Hello UK");
        System.out.println("Hello Brighton");
    }
}

// These lines starting with '//' are COMMENTS
// They are not part of the program itself, but just note
// They don't have to follow Java program rules
// In fact Java when Java sees '//' itjust igonres anythi
```

Class compiled - no syntax errors

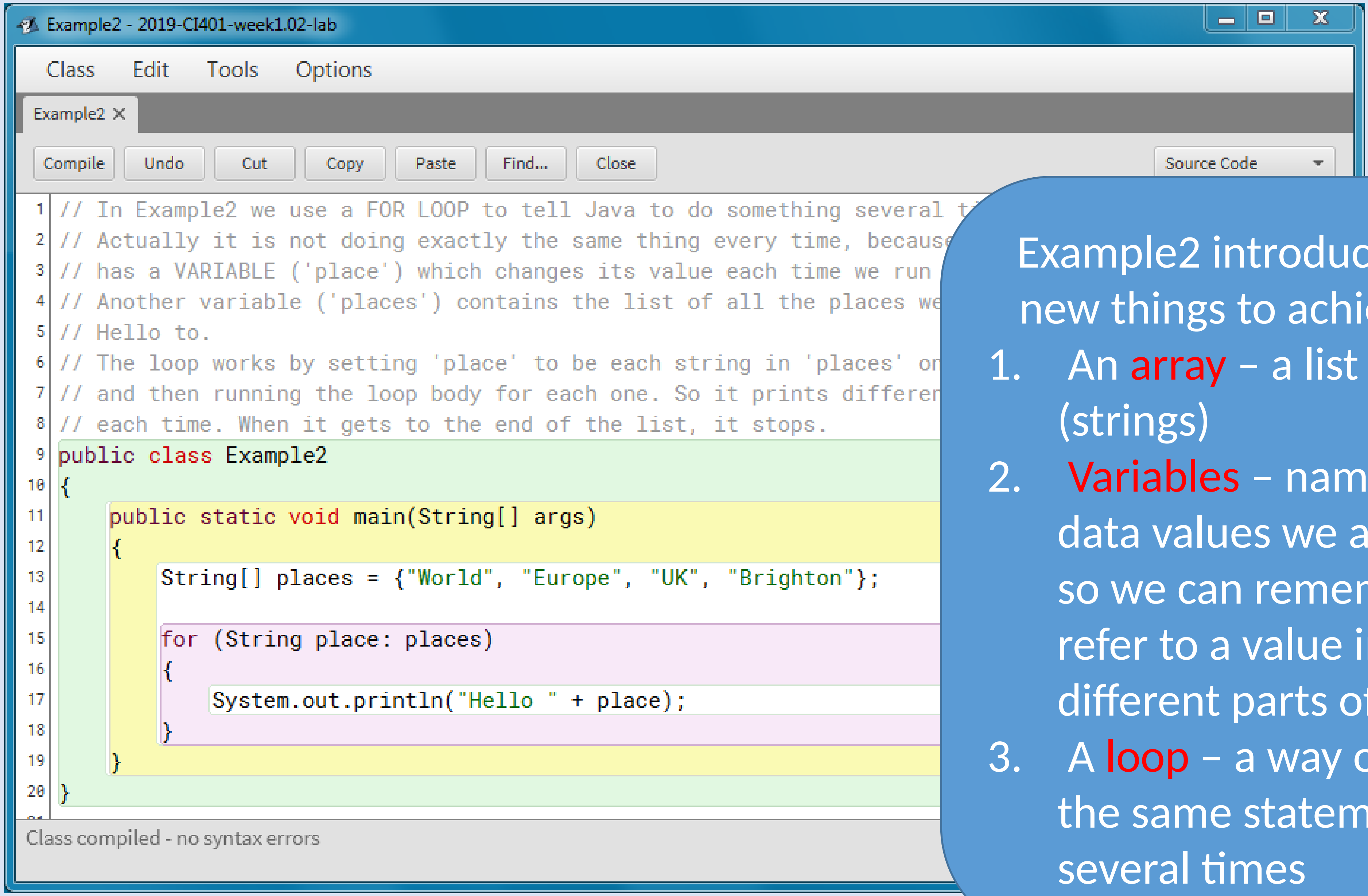
Here is Example 1 from the lab project. It is the same as Example2 from last week except that we have added COMMENTS. These are the lines starting with // - Java ignores them completely, so we can add notes and instructions about the code we are writing



```
// This is just a repeat of Example2 from week1.01
// It is a basic 'Hello world' progrm, but with four instructions instead of just one
// (you can have as many as you like, and they are run in the order given)
public class Example1
{
    public static void main(String[] args)
    {
        System.out.println("Hello World");
        System.out.println("Hello Europe");
        System.out.println("Hello UK");
        System.out.println("Hello Brighton");
    }
}

// These lines starting wi
// They are not part of th
// They don't have to foll
// In fact Java when Java
```

Class compiled - no syntax errors

As we said last week, we have asked the computer to print four things instead of just one, but this is not a great way to do this. Every step has to be spelt out, and there's lots of repetition. Wouldn't it be better to say
'here is a list of places, say "Hello" to each one' ?

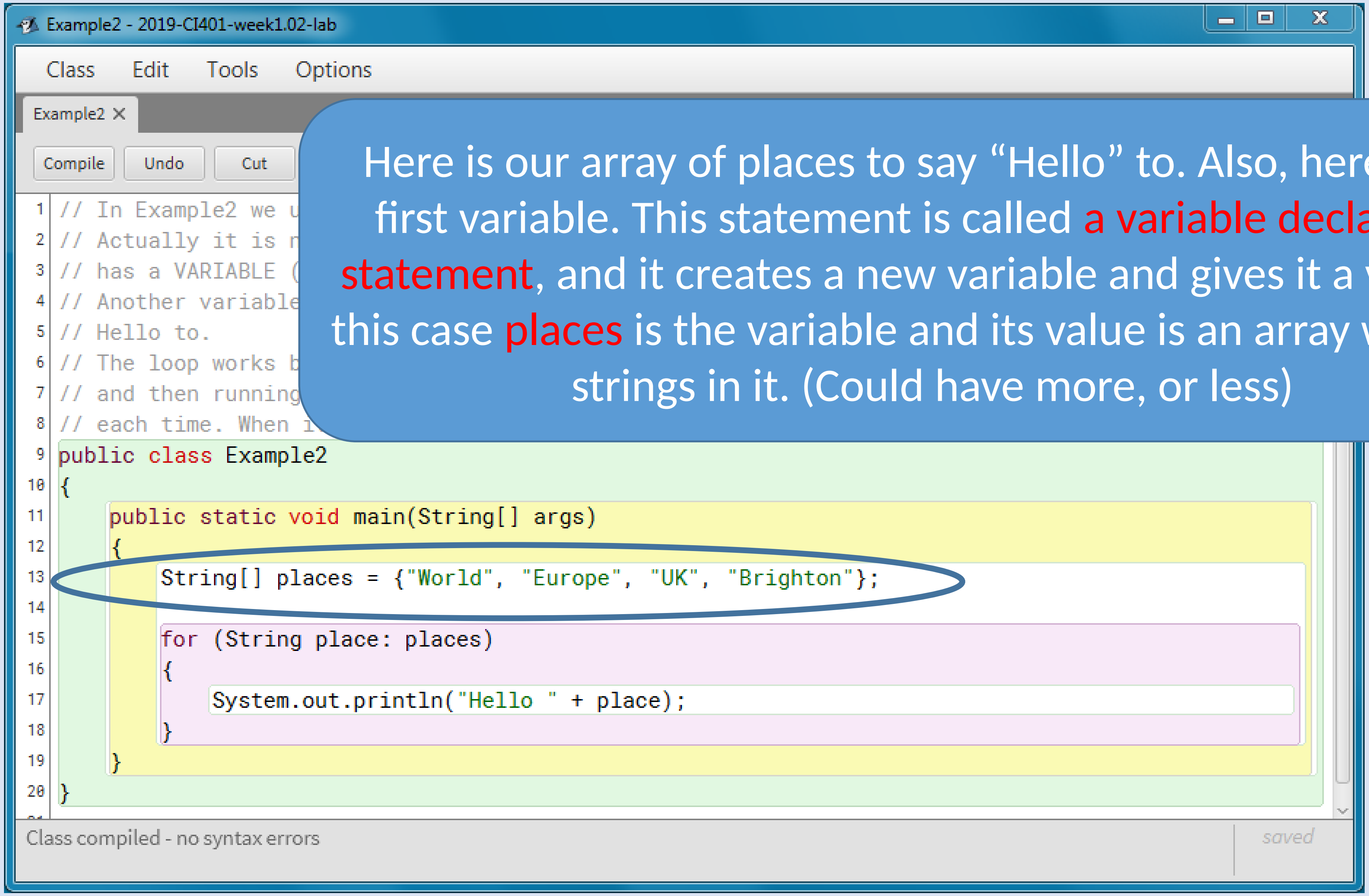

Example2 - 2019-CI401-week1.02-lab
Class Edit Tools Options
Example2
Compile Undo Cut Copy Paste Find... Close
Source Code

```
// In Example2 we use a FOR LOOP to tell Java to do something several ti
// Actually it is not doing exactly the same thing every time, becaus
// has a VARIABLE ('place') which changes its value each time we run
// Another variable ('places') contains the list of all the places we
// Hello to.
// The loop works by setting 'place' to be each string in 'places' on
// and then running the loop body for each one. So it prints differen
// each time. When it gets to the end of the list, it stops.
public class Example2
{
    public static void main(String[] args)
    {
        String[] places = {"World", "Europe", "UK", "Brighton"};

        for (String place: places)
        {
            System.out.println("Hello " + place);
        }
    }
}
```

Class compiled - no syntax errors

Example2 introduces three new things to achieve this:

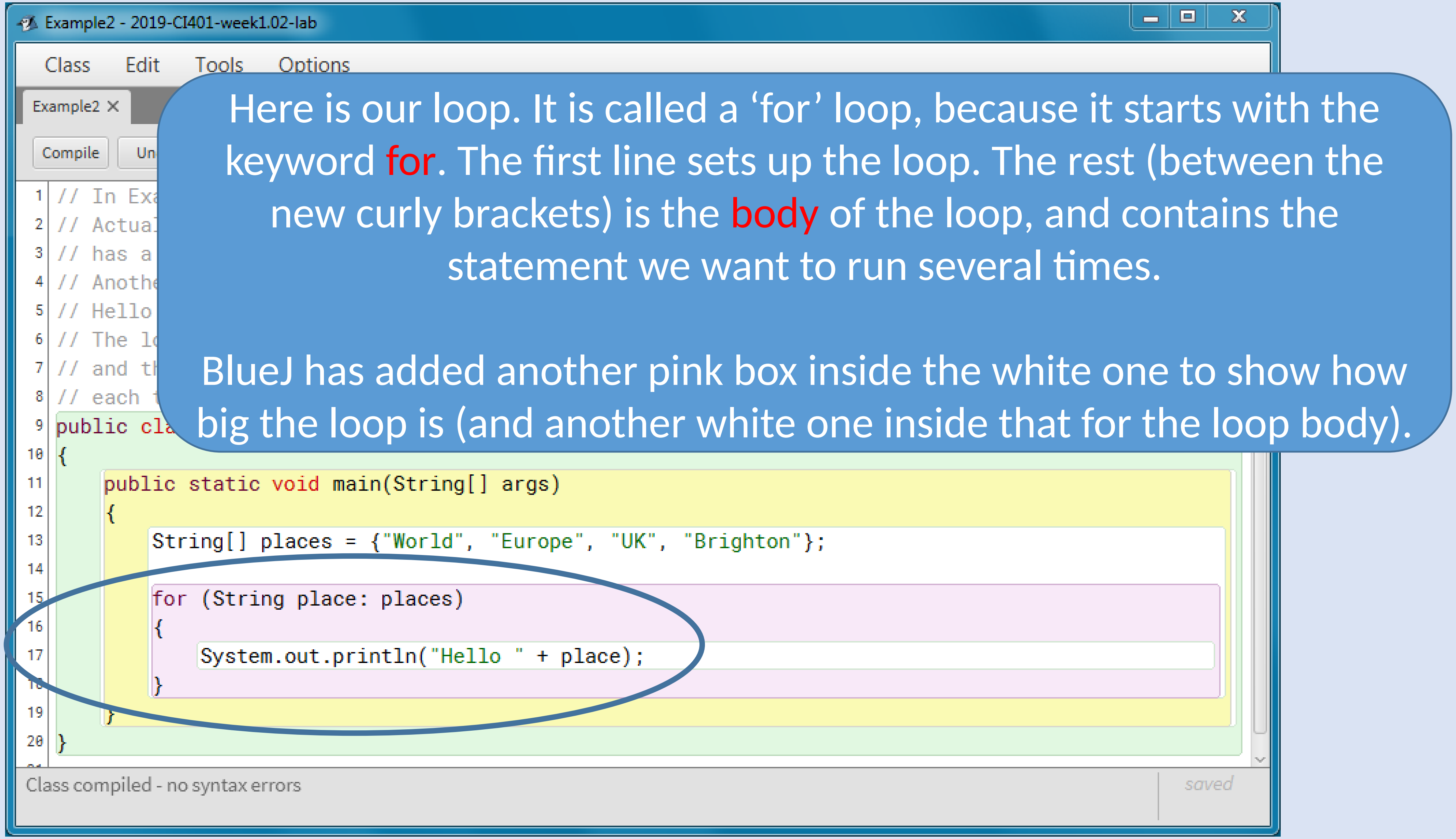
1. An array – a list of places (strings)
2. Variables – names for data values we are using, so we can remember and refer to a value in different parts of the code
3. A loop – a way of running the same statements several times



```
// In Example2 we u
// Actually it is n
// has a VARIABLE (
// Another variable
// Hello to.
// The loop works b
// and then running
// each time. When i
public class Example2
{
    public static void main(String[] args)
    {
        String[] places = {"World", "Europe", "UK", "Brighton"};

        for (String place: places)
        {
            System.out.println("Hello " + place);
        }
    }
}
```

Class compiled - no syntax errors

Here is our array of places to say "Hello" to. Also, here is our first variable. This statement is called a variable declaration statement, and it creates a new variable and gives it a value. In this case places is the variable and its value is an array with four strings in it. (Could have more, or less)



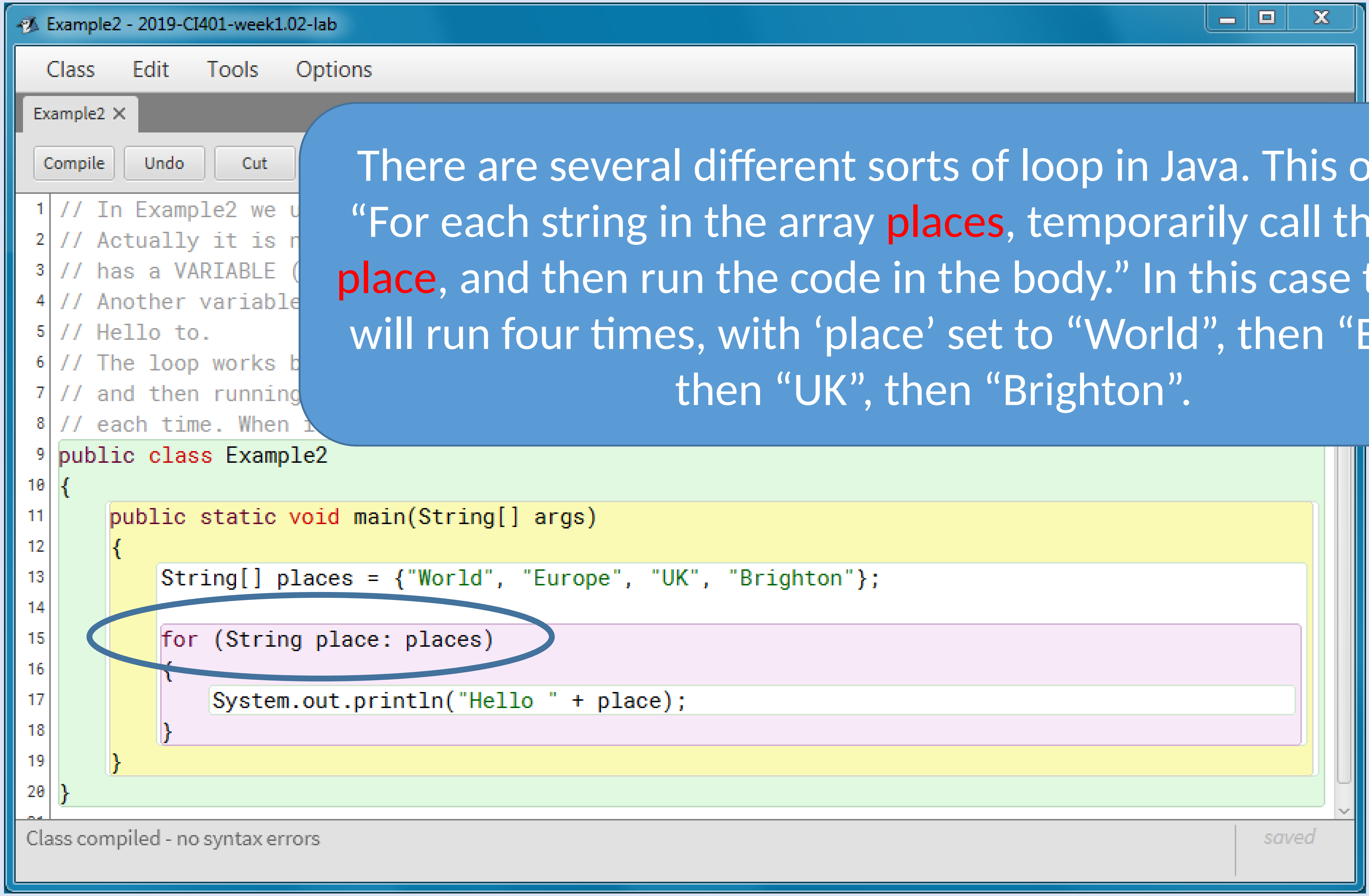
```
// In Exa
// Actua
// has a
// Anothe
// Hello
// The lo
// and th
// each t
public cla
{
    public static void main(String[] args)
    {
        String[] places = {"World", "Europe", "UK", "Brighton"};

        for (String place: places)
        {
            System.out.println("Hello " + place);
        }
    }
}
```

Class compiled - no syntax errors

saved

Here is our loop. It is called a ‘for’ loop, because it starts with the keyword for. The first line sets up the loop. The rest (between the new curly brackets) is the body of the loop, and contains the statement we want to run several times.

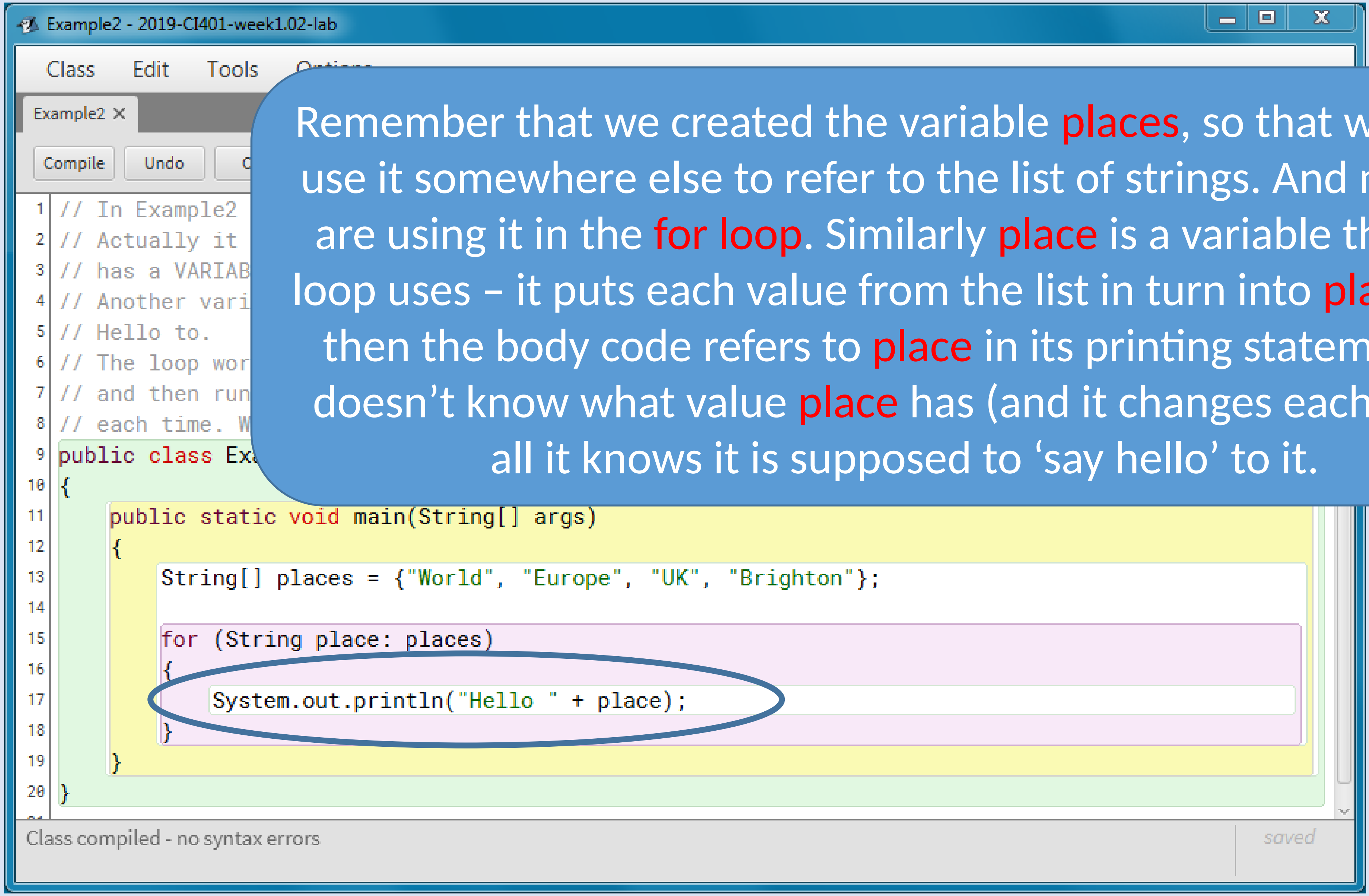
BlueJ has added another pink box inside the white one to show how big the loop is (and another white one inside that for the loop body).



```
// In Example2 we u
// Actually it is n
// has a VARIABLE (
// Another variable
// Hello to.
// The loop works b
// and then running
// each time. When i
public class Example2
{
    public static void main(String[] args)
    {
        String[] places = {"World", "Europe", "UK", "Brighton"};

        for (String place: places)
        {
            System.out.println("Hello " + place);
        }
    }
}
```

Class compiled - no syntax errors

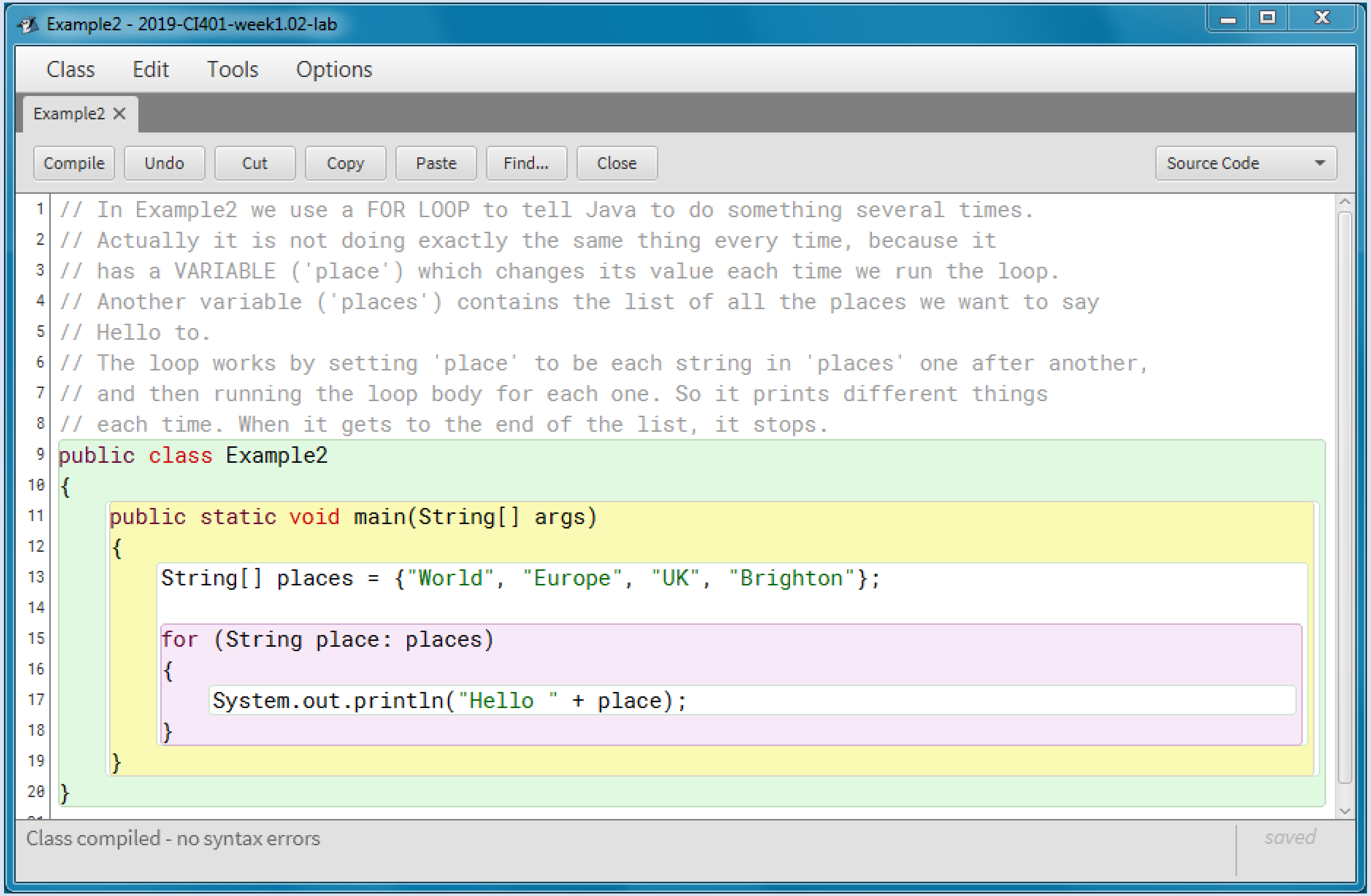
There are several different sorts of loop in Java. This one says "For each string in the array places, temporarily call that thing place, and then run the code in the body." In this case the body will run four times, with 'place' set to "World", then "Europe", then "UK", then "Brighton".



```
// In Example2
// Actually it
// has a VARIAB
// Another vari
// Hello to.
// The loop wor
// and then run
// each time. W
public class Exa
{
    public static void main(String[] args)
    {
        String[] places = {"World", "Europe", "UK", "Brighton"};

        for (String place: places)
        {
            System.out.println("Hello " + place);
        }
    }
}
```

Class compiled - no syntax errors

Remember that we created the variable places, so that we could use it somewhere else to refer to the list of strings. And now we are using it in the for loop. Similarly place is a variable that the loop uses – it puts each value from the list in turn into place, and then the body code refers to place in its printing statement. It doesn't know what value place has (and it changes each time), all it knows it is supposed to 'say hello' to it.

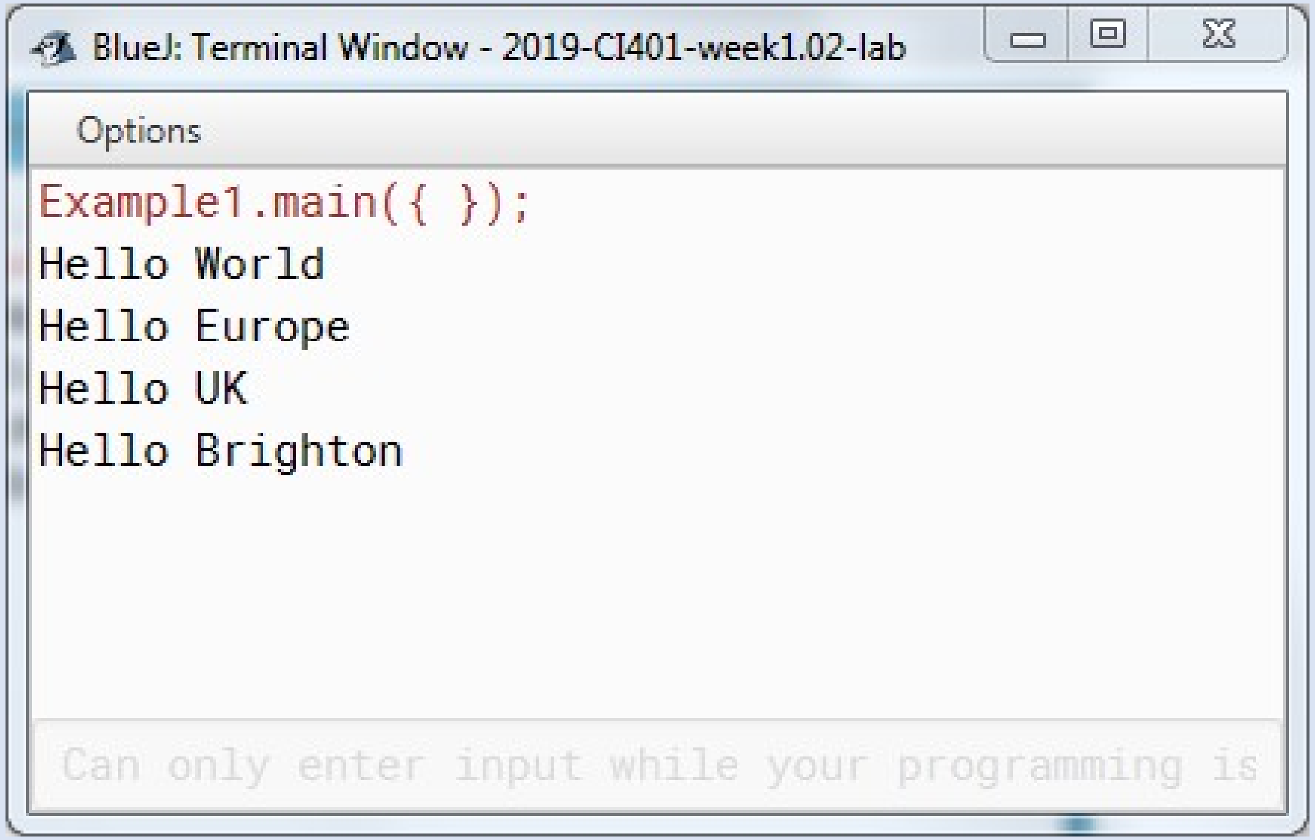


```
// In Example2 we use a FOR LOOP to tell Java to do something several times.
// Actually it is not doing exactly the same thing every time, because it
// has a VARIABLE ('place') which changes its value each time we run the loop.
// Another variable ('places') contains the list of all the places we want to say
// Hello to.
// The loop works by setting 'place' to be each string in 'places' one after another,
// and then running the loop body for each one. So it prints different things
// each time. When it gets to the end of the list, it stops.
public class Example2
{
    public static void main(String[] args)
    {
        String[] places = {"World", "Europe", "UK", "Brighton"};

        for (String place: places)
        {
            System.out.println("Hello " + place);
        }
    }
}
```

Class compiled - no syntax errors

saved

Example2 output


BlueJ: Terminal Window - 2019-CI401-week1.02-lab
Options

```
Example1.main({ });
Hello World
Hello Europe
Hello UK
Hello Brighton
```

Can only enter input while your programming is

Choices

Making choices in a program

- We have seen how in a program we can specify **sequences** of instructions, and create **loops** to run the same instructions more than once.
- Now we will look at how to **make choices** about alternative sets of instructions to run in different circumstances
- The simplest way to do this is with an *if statement*.

Simple if statements

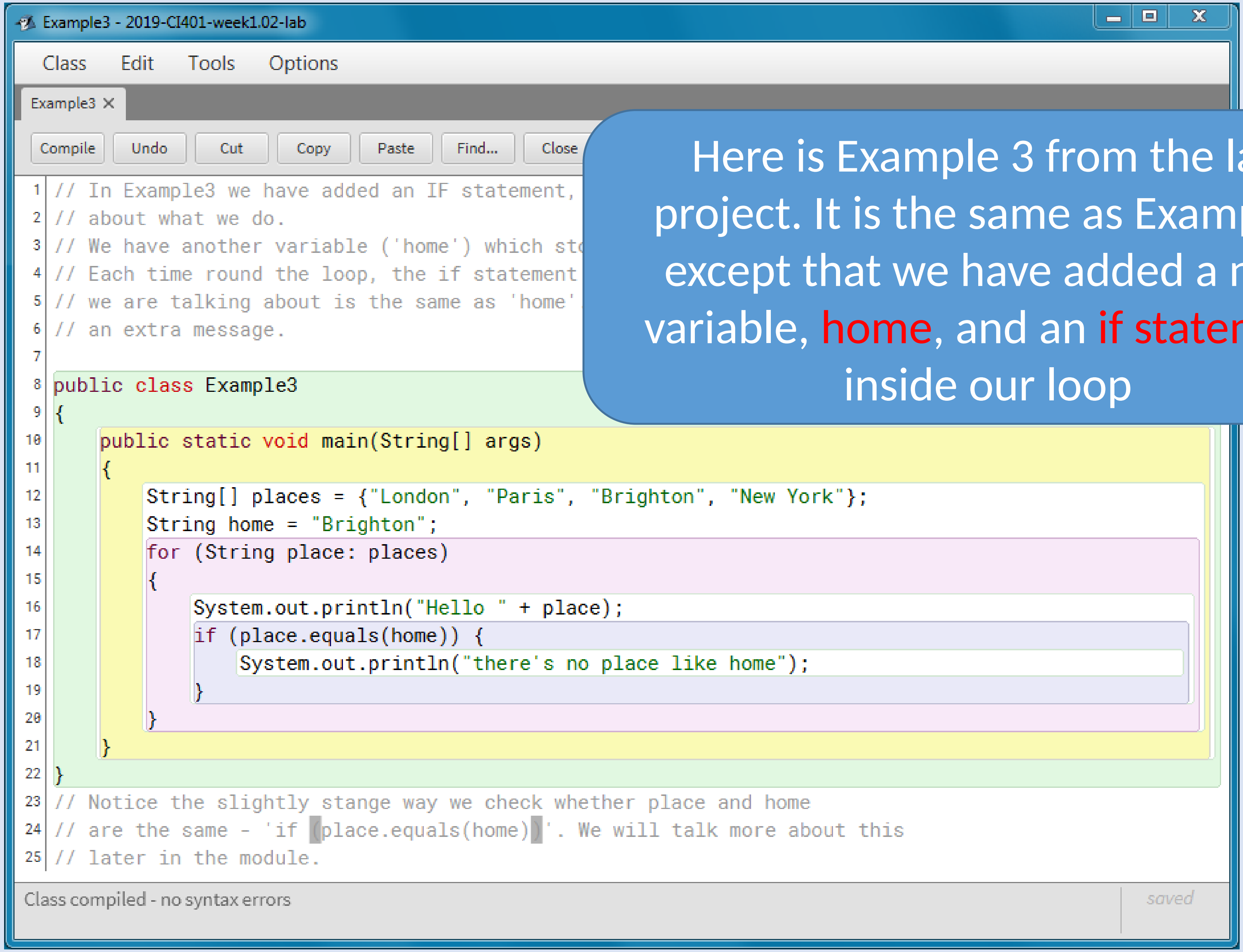
- The simplest form of an if statement looks like this:

```
if ( test )
{
     statements to run if the test is true
}
```

- Like a for loop, this starts with the keyword if, then has a bit in round brackets, followed by some statements in curly brackets
- The way to read it is ‘if the test is true, then run the statements in the curly brackets, otherwise do nothing’

Tests in if statements

- The test part of an if statement is something that can be **true** or **false**
- One common kind of test in programming is to test whether two things are the same or not
- This is not very interesting when the two things are just values – testing whether 2 equals 2, or "Brighton" equals "Hove" is pointless because we know the answer in advance
- It becomes useful when one or both of the two things are variables (so we don’t actually know what their value is in advance)
- So a test such as age equals 18 or home equals "London" will be true or false depending on the value of variables age or home, which may be set somewhere else in the program

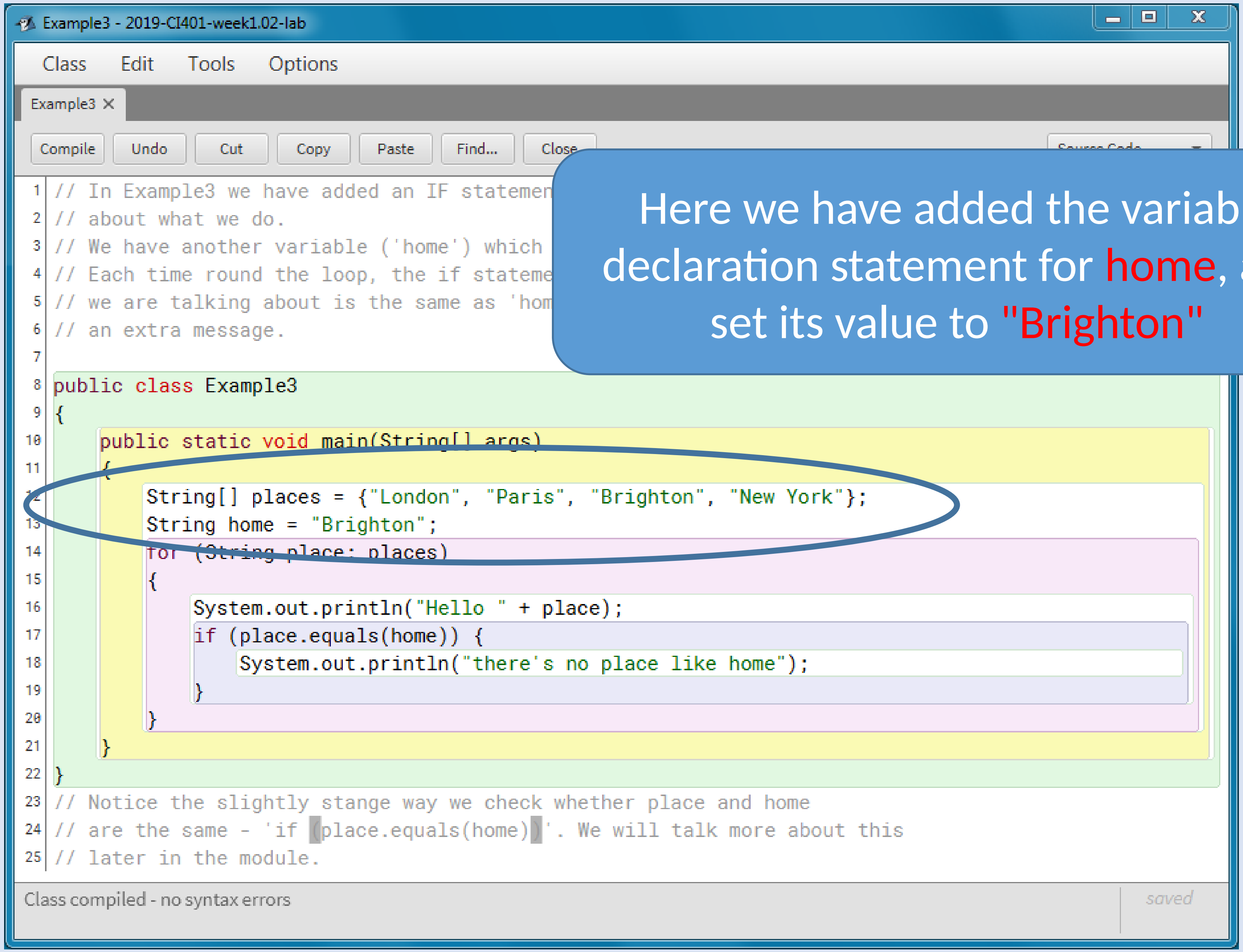

Example3 - 2019-CI401-week1.02-lab
Class Edit Tools Options
Example3
Compile Undo Cut Copy Paste Find... Close

```
// In Example3 we have added an IF statement,
// about what we do.
// We have another variable ('home') which st
// Each time round the loop, the if statement
// we are talking about is the same as 'home'
// an extra message.

public class Example3
{
    public static void main(String[] args)
    {
        String[] places = {"London", "Paris", "Brighton", "New York"};
        String home = "Brighton";
        for (String place: places)
        {
            System.out.println("Hello " + place);
            if (place.equals(home)) {
                System.out.println("there's no place like home");
            }
        }
    }
}
// Notice the slightly stange way we check whether place and home
// are the same - 'if (place.equals(home))'. We will talk more about this
// later in the module.
```

Class compiled - no syntax errors

saved

Here is Example 3 from the lab project. It is the same as Example2 except that we have added a new variable, home, and an if statement inside our loop



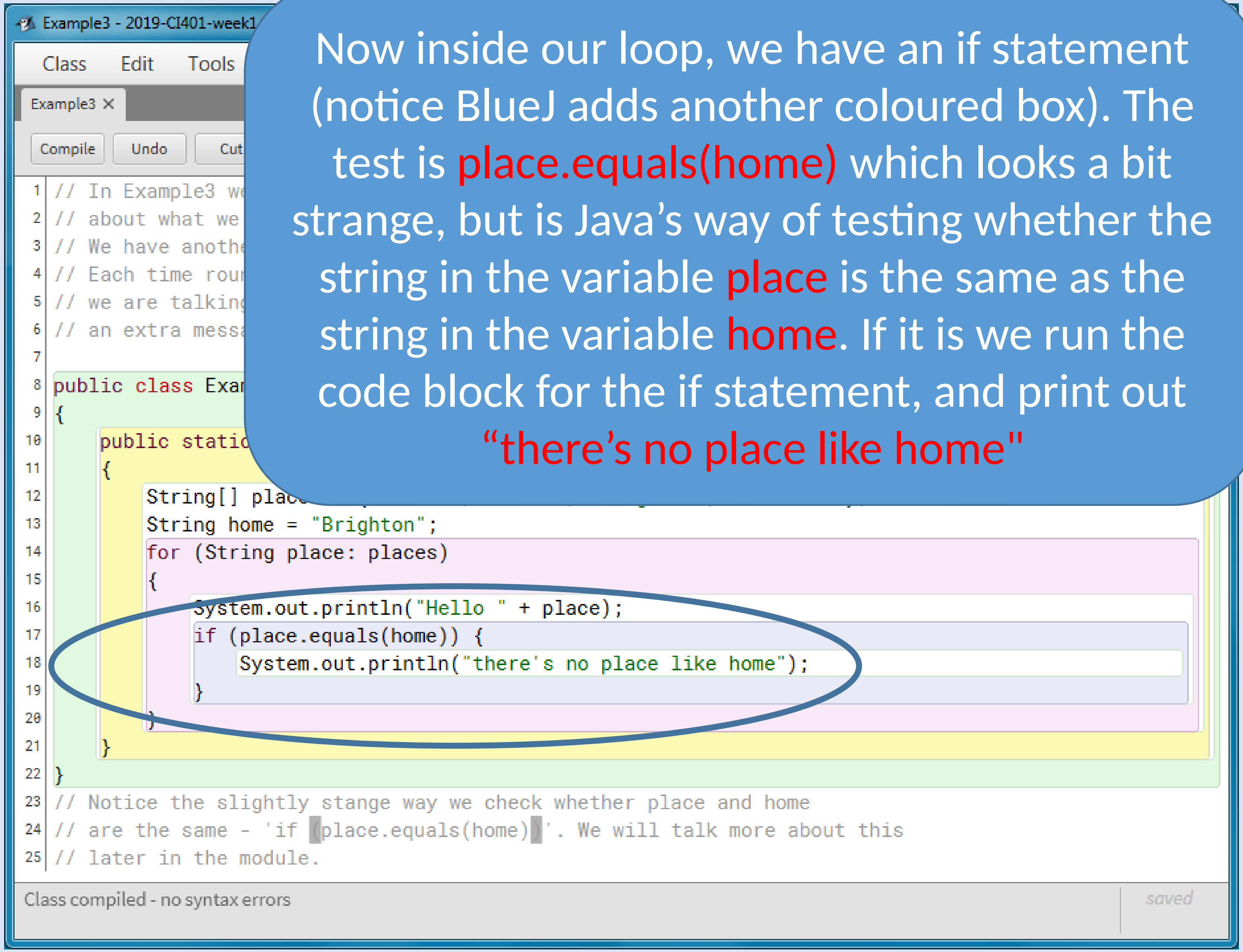
```
// In Example3 we have added an IF statemen
// about what we do.
// We have another variable ('home') which
// Each time round the loop, the if stateme
// we are talking about is the same as 'hom
// an extra message.

public class Example3
{
    public static void main(String[] args)
    {
        String[] places = {"London", "Paris", "Brighton", "New York"};
        String home = "Brighton";
        for (String place: places)
        {
            System.out.println("Hello " + place);
            if (place.equals(home)) {
                System.out.println("there's no place like home");
            }
        }
    }
}
// Notice the slightly stange way we check whether place and home
// are the same - 'if (place.equals(home))'. We will talk more about this
// later in the module.
```

Class compiled - no syntax errors

saved

Here we have added the variable declaration statement for home, and set its value to "Brighton"

Now inside our loop, we have an if statement (notice BlueJ adds another coloured box). The test is place.equals(home) which looks a bit strange, but is Java’s way of testing whether the string in the variable place is the same as the string in the variable home. If it is we run the code block for the if statement, and print out “there’s no place like home"

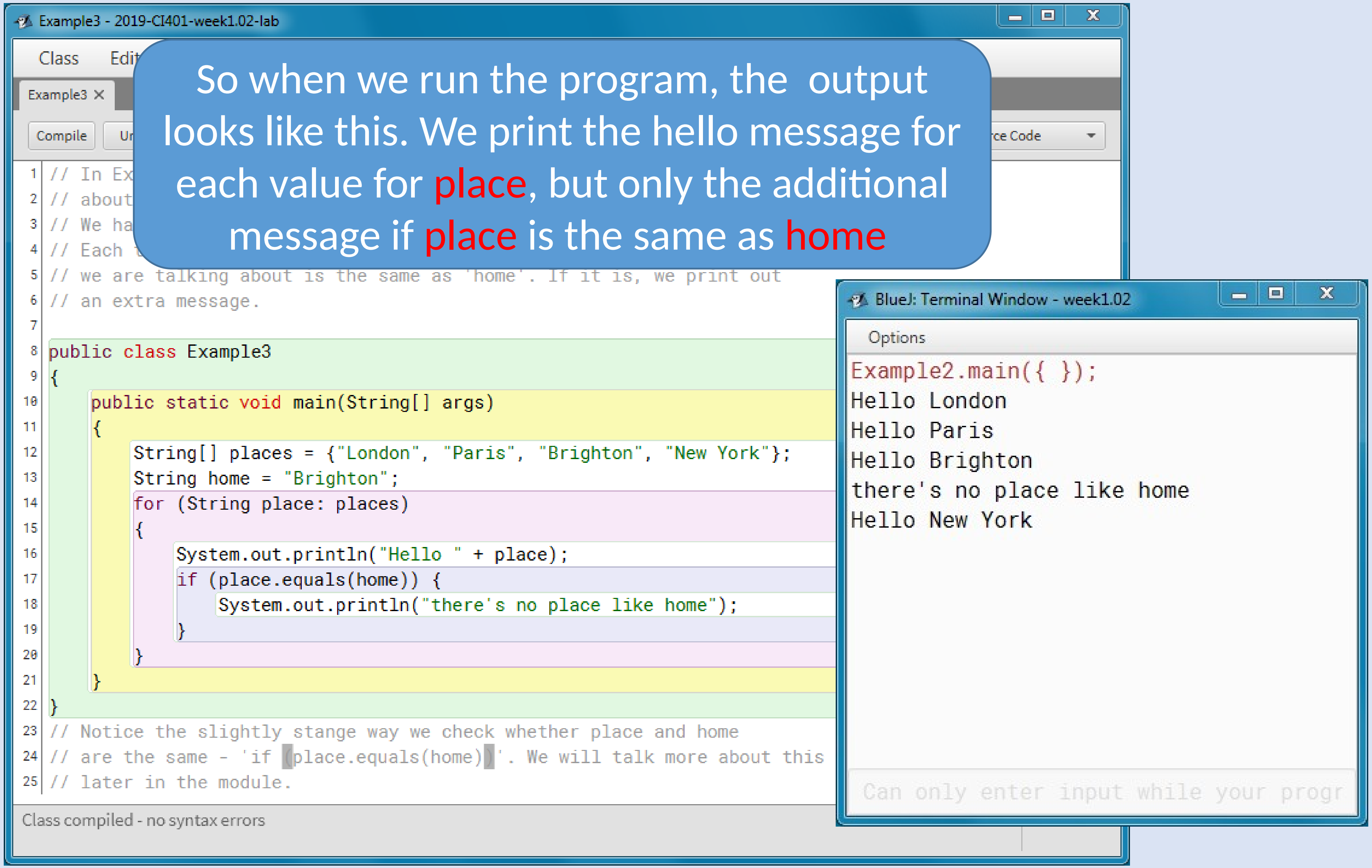


```
// In Example3 we
// about what we
// We have anothe
// Each time rour
// we are talking
// an extra messa

public class Exar
{
    public static
    {
        String[] place
        String home = "Brighton";
        for (String place: places)
        {
            System.out.println("Hello " + place);
            if (place.equals(home)) {
                System.out.println("there's no place like home");
            }
        }
    }
}
// Notice the slightly stange way we check whether place and home
// are the same - 'if (place.equals(home))'. We will talk more about this
// later in the module.
```

Class compiled - no syntax errors

saved

So when we run the program, the output looks like this. We print the hello message for each value for place, but only the additional message if place is the same as home


Example3 - 2019-CI401-week1.02-lab
Class
Edit
Example3
Compile
ce Code
BlueJ: Terminal Window - week1.02
Options

```
// In Ex
// about
// We ha
// Each t
// we are talking about is the same as 'home'. If it is, we print out
// an extra message.

public class Example3
{
    public static void main(String[] args)
    {
        String[] places = {"London", "Paris", "Brighton", "New York"};
        String home = "Brighton";
        for (String place: places)
        {
            System.out.println("Hello " + place);
            if (place.equals(home)) {
                System.out.println("there's no place like home");
            }
        }
    }
}
// Notice the slightly stange way we check whether place and home
// are the same - 'if (place.equals(home))'. We will talk more about this
// later in the module.
```

Class compiled - no syntax errors

```
Example2.main({ });
Hello London
Hello Paris
Hello Brighton
there's no place like home
Hello New York
```

Can only enter input while your progr

Other styles of if statement

```
if ( test ) {
    statements to run if the test is true
} else {
    statements to run if the test is false
}
```

- This versions has two blocks of code – one to do if the test is true, and the other (known as the else clause) to do if the test is false.
- Notice all the round and curly brackets – it's important to get them right!
- The whole thing is one statement – there may be statements before and after it. All it controls is the choice between these two blocks of code

Other styles of if statement

- Sometimes you want to test other things if the first test fail. You can chain multiple if statements together like this:

```
if ( test1 ) {
     statements to run if test1 is true
} else if (test2 ) {
     statements to run if test2 is true
} else if (test3 ) {
     statements to run if test3 is true
} else {
     statements to run if all tests are false
}
```

Combining statements

- For loops and if statements can each be thought of as single instructions
- You can combine them with each other as much as you like
- It’s absolutely fine to include a whole for loop or if statement inside the code block of another one – the matching curly brackets for each block make sure everything makes sense
- BlueJ helps you by indenting code which is ‘inside’ another block of code (and changing its box colour).
- It’s a good idea to be clear with your indenting, because getting a bracket in the wrong place can make it hard to spot errors
- BlueJ can do the indenting automatically for you (Edit ☾ Auto-Layout)

Summary

Core principles of programming

- We have learned about three basic coding ideas in Java which are know as the Core principles of programming:
  - Sequence – the ability to run instructions one after another
  - Selection – the ability to select which instructions to run
  - Iteration – the ability to repeat instructions multiple times
- Using just these three things, we can write almost any program imaginable

Key ideas

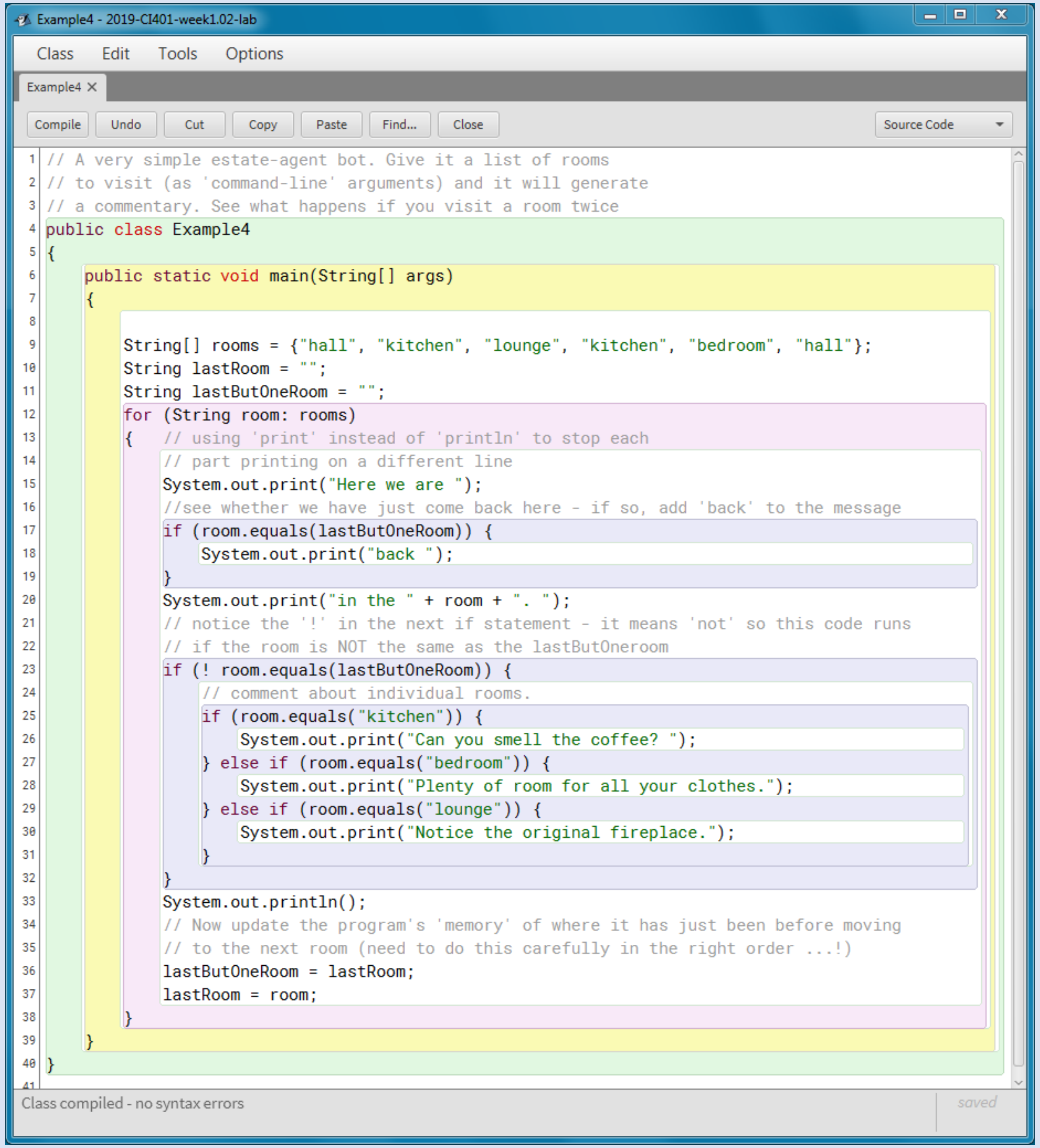
- Variables – names for values we use or calculate in a program
- For loops – complex statements for repeating (iterating) instructions
- If statements – complex statements for making choices
- Keywords (for, if, else) – words which have special meaning to Java
- Tests – things which can be true or false, for making choices
- Combining elements – curly brackets define the structure of complex statements

Challenge example

For more experienced coders

Example4
The estate agent bot

- Example4 uses a for loop, a few variables and if statements to make a simple ‘Estate Agent’ bot
- It takes you on a tour of a house, describing each room
- It remembers when you go back to a room you just came from, and adjusts its comments appropriately



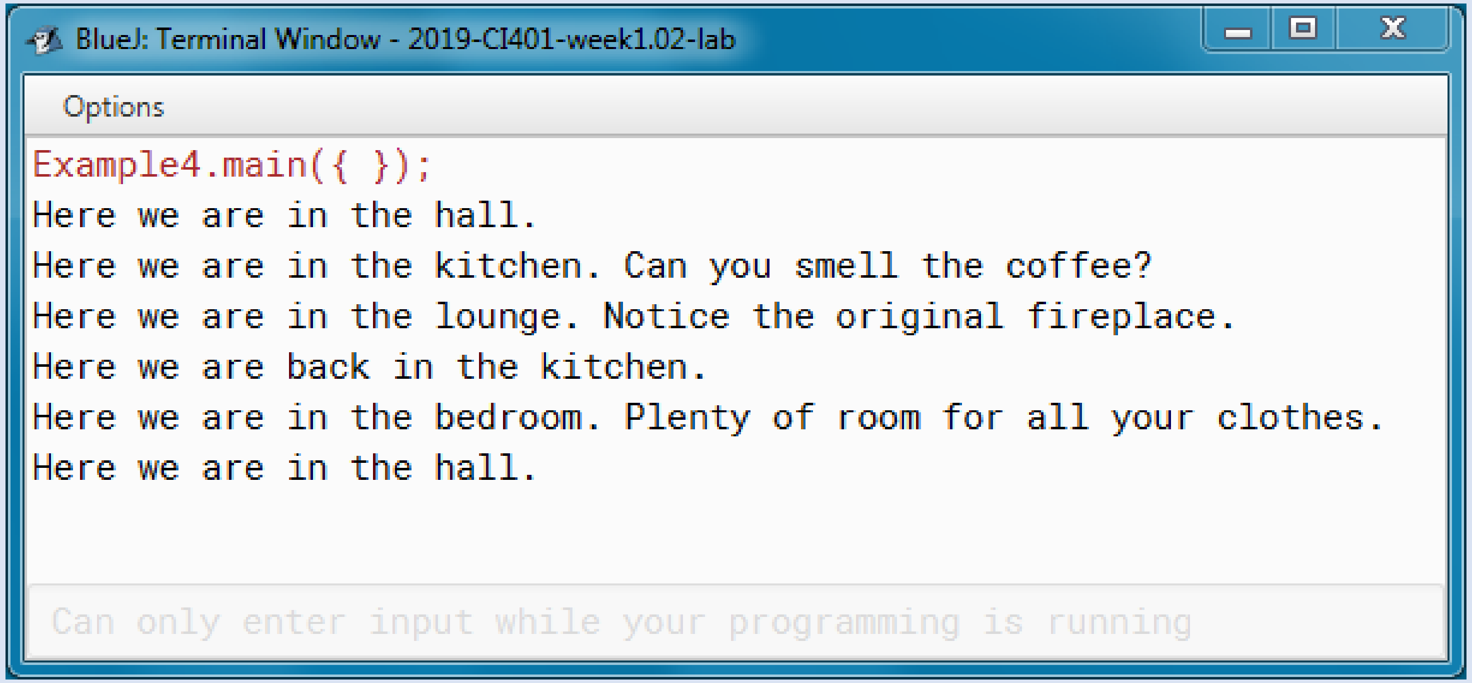
```java
// A very simple estate-agent bot. Give it a list of rooms
// to visit (as 'command-line' arguments) and it will generate
// a commentary. See what happens if you visit a room twice
public class Example4
{
    public static void main(String[] args)
    {

        String[] rooms = {"hall", "kitchen", "lounge", "kitchen", "bedroom", "hall"};
        String lastRoom = "";
        String lastButOneRoom = "";
        for (String room: rooms)
        {   // using 'print' instead of 'println' to stop each
            // part printing on a different line
            System.out.print("Here we are ");
            //see whether we have just come back here - if so, add 'back' to the message
            if (room.equals(lastButOneRoom)) {
                System.out.print("back ");
            }
            System.out.print("in the " + room + ". ");
            // notice the '!' in the next if statement - it means 'not' so this code runs
            // if the room is NOT the same as the lastButOneroom
            if (! room.equals(lastButOneRoom)) {
                // comment about individual rooms.
                if (room.equals("kitchen")) {
                    System.out.print("Can you smell the coffee? ");
                } else if (room.equals("bedroom")) {
                    System.out.print("Plenty of room for all your clothes.");
                } else if (room.equals("lounge")) {
                    System.out.print("Notice the original fireplace.");
                }
            }
            System.out.println();
            // Now update the program's 'memory' of where it has just been before moving
            // to the next room (need to do this carefully in the right order ...!)
            lastButOneRoom = lastRoom;
            lastRoom = room;
        }
    }
}
```

Class compiled - no syntax errors

Example4
The estate agent bot

- Here's the output
- Notice the different messages for each room
- When we return to the kitchen, two things change – it says we have come 'back', and it doesn't repeat the description.

- In the last line, why does it not say we are 'back' in the hall?


BlueJ: Terminal Window - 2019-CI401-week1.02-lab
Options

```
Example4.main({ });
Here we are in the hall.
Here we are in the kitchen. Can you smell the coffee?
Here we are in the lounge. Notice the original fireplace.
Here we are back in the kitchen.
Here we are in the bedroom. Plenty of room for all your clothes.
Here we are in the hall.
```

Can only enter input while your programming is running

Week 1.02 Labs

Lab exercises – BlueJ

- Create a folder for this week’s work on your S: drive eg at S:\CI401\week1.02
- Download BlueJ project week1.02-lab.jar from StudentCentral into this folder
- There is also a .zip version – useful for Eclipse users and Macs.
- Open BlueJ on your computer and create a new project from the jar/zip file in your new folder
- BlueJ will show you a folder full of Example files and Lab exercises

Lab exercises – coding

- Open each of the example files (double click them) and look at the code. Try to understand what each one does (look at the lecture slides too), and then compile and run it to see if you were right
- Open each of the lab files and follow the instructions at the top to edit the code to do something new. Then compile and run it to see if it works.
- Lab 4 is a challenge lab – don’t worry if you can’t do it
- Remember to save your work to your S: drive before finishing.
- If you want to access your labs at home as well, copy the week1.02 folder to your O: drive

Lab exercises – we are here to help!

- **If you get stuck, ask for help!**
- **Even if you don’t get stuck, talk to us!**