Started on	Monday, 4 March 2019, 12:49 AM
State	Finished
Completed on	Wednesday, 6 March 2019, 1:06 AM
Time taken	2 days
Marks	6.90/7.00
Grade	**98.57** out of 100.00

Information

General Lab Information

This lab should take you about one week to complete.

Goals:

- Gain first experience in designing, implementing, running and modifying Java programs.
- Ensure that you are able to use Eclipse for your Java development projects.

You may not be able to complete the whole exercise during the lab session. However, you should try to complete it before your next lab session as study. You are welcome to use the Learn forum or to drop in to another lab session if you have questions. Some exercises not only include concepts discussed in the previous week, but also include ideas and concepts that are covered in the lectures of the week of the lab.

If you have any issues, have a chat to A or B, your friendly tutors.

Note, labs aren't worth any course credit, however we highly recommend you complete all labs. Labs will help you prepare for the midterm test and assignment.

Response history

Step	Time	Action	State
1	4/03/19, 00:49	Started	
2	6/03/19, 00:17	Seen	
3	**6/03/19, 01:06**	**Attempt finished**	

Information

HelloWorld Class

In Java, all programs come in units called *classes*, which are a collection of related code typically relating to some sort of real world object. Think astronauts and rocket ships: code that powers an astronaut would belong in an Astronaut class, and not a RocketShip class.

The basic structure of a class looks like this:

```
public class HelloWorld {
        public static void main(String[] args) {
                System.out.println("Hello, World!");
        }
}
```

The first line tells us the *class name*, which is "HelloWorld" in this example. Note that in Java all classes start with a capital letter, and the 'W' is also capitalized since we follow camel case.

The next line is the main method. This is the method that Java calls first in any program. Classes don't have to have main methods, but classes that you wish to run as standalone programs do. We will talk about this later.

As you can see, Java has some keywords in front of methods. The "public" part tells us that the current class and any other class may call this method. If it were to be made private, we can call the method from our class, but not from other classes. We will discuss this in more detail in the lectures.

"static" means that we do not need to *instantiate* an object to call the method. This can be helpful for basic methods that do not really need to be a part of a class, such as methods of a class that do not modify any variables, but this is typically not normal. Please ensure that main() is your only static method. Again, we will talk more about this in the lecture.

Finally, "void" tells us that the method does not return anything. If it were to return something, then we would use the type of whatever we return instead of "void". We will talk about data types in the lecture. Unlike Python, Java forces us to specify the data types of variables and return values of methods.

Enough chatter, lets run some code.

Response history

Step	Time	Action	State
1	4/03/19, 00:49	Started	
2	6/03/19, 00:17	Seen	
3	**6/03/19, 01:06**	**Attempt finished**	

Information

Running Java

Open up a text editor, something like gedit, Atom, or Kate will do. (We will get to IDE's shortly.)

Copy and paste the above program into the text editor, and save the file as HelloWorld.java to your home folder.

The name is pretty important. All file names need to match the class name, otherwise Java will get confused.

Open up a terminal, by searching for terminal or by pressing ctrl-alt-t. The terminal is already open to your home folder, so if you saved your file elsewhere, move it to your home folder, or better yet, learn how to use the "cd" command in Linux.

Java is a compiled language, so we need to compile the source code into *bytecode*. As we have learned in the lecture, the bytecode is then executed in the Java virtual machine, which gives you its "write once, run everywhere" philosophy.

The commands we will be using for this lab are:

```
java <classname>
javac <source file>.java
```

Compile your program by typing (in the terminal)

```
javac HelloWorld.java
```

and run your program by typing (in the terminal)

```
java HelloWorld
```

Hopefully you have successfully run your first Java program. If you now type "ls" into your terminal, you will see two files. HelloWorld.java and HelloWorld.class. The .class file is the compiled bytecode. The java command searches all files in the directory for a class called "HelloWorld", and finds one in HelloWorld.class and runs it.

Response history

Step	Time	Action	State
1	4/03/19, 00:49	Started	
2	6/03/19, 00:17	Seen	
3	**6/03/19, 01:06**	**Attempt finished**	

Question **1**

Correct

Mark 1.00 out of 1.00

Java Commands

Match the following:

The file extension for Java bytecode	.class ✓
The file extension for Java source code	.java ✓
The program/command which runs a class	java ✓
The program/command which compiles source code into bytecode	javac ✓

Your answer is correct.

The correct answer is: The file extension for Java bytecode → .class, The file extension for Java source code → .java, The program/command which runs a class → java, The program/command which compiles source code into bytecode → javac

Correct

Marks for this submission: 1.00/1.00.

Response history

Step	Time	Action	State	Marks
1	4/03/19, 00:49	Started	Not complete	
2	6/03/19, 00:17	Submit: The file extension for Java bytecode -> .class; The file extension for Java source code -> .java; The program/command which runs a class -> java; The program/command which compiles source code into bytecode -> javac	Correct	1.00
3	**6/03/19, 01:06**	**Attempt finished**	**Correct**	**1.00**

Question **2**

Correct

Mark 1.00 out of 1.00

StarmanSays Class

Starman thinks that outputting "Hello, World!" is kind of lame, since when you are floating majestically in outer space in your red Tesla Roadster, earth is the last thing on your mind. Change the HelloWorld class to be called StarmanSays, and output the string "Don't Panic!" instead of "Hello, World!".

Make sure to change the filename and terminal commands when you compile to test. Before writing and submitting your code in the input field on Quiz Server, you should make sure the code compiles and runs by first writing it in a text editor or IDE and compiling/running it in the terminal (**please do not treat the Quiz Server as your code editor or compiler, this is a bad habit to get into**).

Answer: (penalty regime: 0, 10, 20, ... %)

```
public class StarmanSays {
    public static void main(String[] args) {
        System.out.println("Don't Panic!");
    }
}
```

	Test	Expected	Got	
✓	`String[] strlist = {"Hey Tutor!"};` `StarmanSays.main(strlist);`	`Don't Panic!`	`Don't Panic!`	✓

Passed all tests! ✓

Question author's solution:

```
public class StarmanSays {
    public static void main(String[] args) {
        System.out.println("Don't Panic!");
    }
}
```

Correct

Marks for this submission: 1.00/1.00.

Response history

Step	Time	Action	State	Marks
1	4/03/19, 00:49	Started	Not complete	
2	6/03/19, 00:18	Submit: public class StarmanSays { public static void main(String[] args) { System.out.println("Don't Panic!"); } }	Correct	1.00
3	**6/03/19, 01:06**	**Attempt finished submitting:**	**Correct**	**1.00**

Information

Methods

Starman wants to say more than Don't Panic!, so we will introduce the idea of methods. Methods are just like functions in Python, with some slightly different syntax. Lets look at an example.

```
public class StarmanSays {

        /**
         * Methods are placed above the main method
         * in this section of the program
         */

        public String sayHello() {
                return "Hello";
        }

        public static void main(String[] args) {
                StarmanSays starman = new StarmanSays();   // new instance of StarmanSays
                System.out.println(starman.sayHello());
        }
}
```

First, note that Java has a different syntax for comments. In Java, single line comments are denoted with two forward slashes, while multi line comments start with a /*, and end with a */.

In the example, there are a few new things. For starters, in the main() method, there is a line that creates a new *instance* of StarmanSays. You can have multiple instances of the same class in a program, and will be extremely useful. starman is a variable of type StarmanSays, and a 'new' object is created and placed in the variable.

You call methods by calling the method name on the object it belongs to. In this case, starman is an instantiated object, that has a sayHello() method, so we can call it using starman.sayHello().

Response history

Step	Time	Action	State
1	4/03/19, 00:49	Started	
2	6/03/19, 00:17	Seen	
3	**6/03/19, 01:06**	**Attempt finished**	

Question **3**

Correct

Mark 1.00 out of 1.00

Extending StarmanSays

Add the following three new methods to the StarmanSays class

(**before writing the code, sketch a UML class diagram of your class**).

- sayDontPanic(), which returns the string "Don't Panic!"
- sayJavaCool(), which returns the string "Java is cool"
- sayGoodbyeWorld() which returns the string "Goodbye, World!"
- (Remember to also include the sayHello() method from above)

For example:

Test	Result

`StarmanSays starman = new StarmanSays();`
`System.out.println(starman.sayHello());`
	`Hello`

`StarmanSays starman = new StarmanSays();`
`System.out.println(starman.sayDontPanic());`
	`Don't Panic!`

Answer: (penalty regime: 0, 10, 20, ... %)

```java
public class StarmanSays {

    /**
     * Methods are placed above the main method
     * in this section of the program
     */

    public String sayHello() {
        return "Hello";
    }
    public String sayDontPanic() {
        return "Don't Panic!";
    }
    public String sayJavaCool() {
        return "Java is cool";
    }
    public String sayGoodbyeWorld() {
        return "Goodbye, World!";
    }

    public static void main(String[] args) {
        StarmanSays starman = new StarmanSays();    // new instance of StarmanSays
```

	Test	Expected	Got	

✓	`StarmanSays starman = new StarmanSays();`
	`System.out.println(starman.sayHello());`
		`Hello`	`Hello`	✓

✓	`StarmanSays starman = new StarmanSays();`
	`System.out.println(starman.sayDontPanic());`
		`Don't Panic!`	`Don't Panic!`	✓

✓	`StarmanSays starman = new StarmanSays();`
	`System.out.println(starman.sayGoodbyeWorld());`
		`Goodbye, World!`	`Goodbye, World!`	✓

Passed all tests! ✓

Question author's solution:

```
public class StarmanSays {

        /**
         * Methods are placed above the main method
         * in this section of the program
         */

        public String sayDontPanic() {
            return "Don't Panic!";
        }

        public String sayJavaCool() {
            return "Java is cool";
        }

        public String sayGoodbyeWorld() {
            return "Goodbye, World!";
        }

        public String sayHello() {
                return "Hello";
        }

        public static void main(String[] args) {
                StarmanSays starman = new StarmanSays();
                System.out.println(starman.sayHello());
        }
}
```

Correct

Marks for this submission: 1.00/1.00.

Response history

Step	Time	Action	State	Marks
1	4/03/19, 00:49	Started	Not complete	
2	6/03/19, 00:21	Submit: public class StarmanSays { /** * Methods are placed above the main method * in this section of the program */ public String sayHello() { return "Hello"; } public String sayDontPanic() { return "Don't Panic!"; } public String sayJavaCool() { return "Java is cool"; } public static void main(String[] args) { StarmanSays starman = new StarmanSays(); // new instance of StarmanSays System.out.println(starman.sayHello()); System.out.println(starman.sayDontPanic()); System.out.println(starman.sayJavaCool()); } }	Incorrect	0.00
3	6/03/19, 00:23	Submit: public class StarmanSays { /** * Methods are placed above the main method * in this section of the program */ public String sayHello() { return "Hello"; } public String sayDontPanic() { return "Don't Panic!"; } public String sayJavaCool() { return "Java is cool"; } public String sayGoodbyeWorld() { return "Goodbye, World!"; } public static void main(String[] args) { StarmanSays starman = new StarmanSays(); // new instance of StarmanSays System.out.println(starman.sayHello()); System.out.println(starman.sayDontPanic()); System.out.println(starman.sayJavaCool()); System.out.println(starman.sayGoodbyeWorld()); } }	Correct	1.00
4	**6/03/19, 01:06**	**Attempt finished submitting:**	**Correct**	**1.00**

Information

Variables

Variables are how you can store data inside a class. Their values can be set on a per object basis, so if you have multiple objects of the same class, their variables can be set differently. Let's have a look.

```
public class StarmanSays {

        /**
         * Member variables are normally placed at the top
         * of the class, right below the class name.
         */
        public String hello = "Hello";
        public static String helloStatic = "Hello";
        public static final String HELLO_FINAL = "Hello";

        public String sayHello() {
                return hello;
        }

        public static void main(String[] args) {

                // Make two objects
                StarmanSays starman = new StarmanSays();
                StarmanSays starwoman = new StarmanSays();

                // Say hello
                System.out.println(starman.sayHello());
                // Say hello
                System.out.println(starwoman.sayHello());

                // Change to Goodbye
                starman.hello = "Goodbye";
                // Say goodbye
                System.out.println(starman.sayHello());
                // Say hello
                System.out.println(starwoman.sayHello());

                // Say to the stars
                starman.helloStatic = "To the stars!";
                System.out.println(starman.helloStatic);
                System.out.println(starwoman.helloStatic);

                // Attempt to modify
                starman.HELLO_FINAL = "Goodbye"; // compile error, comment out.
                System.out.println(starman.HELLO_FINAL);
                System.out.println(starwoman.HELLO_FINAL);
        }
}
```

Sketch a UML class diagram of the above class.

There are three variables here. The "hello" variable is a normal string, and has been substituted in sayHello().

The helloStatic variable has been marked static. Static means that if its value is changed in one class, it will be changed in all instances of that class.

Final means that the value cannot be changed. We normally use capital letters for the variable name to show this.

Have a go at running the code above. The output should look like this:

```
$ javac StarmanSays.java
$ java StarmanSays
Hello
Hello
Goodbye
Hello
To the stars!
To the stars!
Hello
Hello
```

Response history

Step	Time	Action	State
1	4/03/19, 00:49	Started	
2	6/03/19, 00:32	Seen	
3	**6/03/19, 01:06**	**Attempt finished**	

Question **4**

Correct

Mark 0.90 out of 1.00

Astronaut Class

Write a new Java class called Astronaut with the following variables and methods.
(before writing the code, draw a UML class diagram for this class)

Variables:

- name which is of type String, default "Starman"
- age which is of type int, default 1
- rocketShip which is of type String, default "Falcon Heavy"
- poweredBy which is of type String, default "Electricity"
- inSpace variable which is of type boolean, default true

Methods:

- void printName() which uses System.out.println to output the line: "Hello, my name is <name>"
- void printAge() which uses System.out.println to output the line: "I am <age> years old"
- void printRocket() which uses System.out.println to output the line "My rocket is <rocketShip> and it is powered by <poweredBy>"
- boolean isInSpace() which returns the value of inSpace

For example:

Test	Result

`Astronaut starman = new Astronaut();`
`starman.printName();`
`starman.printAge();`
`starman.printRocket();`
`System.out.println(starman.isInSpace());`
	`Hello, my name is Starman`
	`I am 1 years old`
	`My rocket is Falcon Heavy and it is powered by Electricity`
	`true`

Answer: (penalty regime: 0, 10, 20, ... %)

```java
public class Astronaut {

    /**
     * Member variables are normally placed at the top
     * of the class, right below the class name.
     */
    public String name = "Starman";
    public int age = 1;
    public String rocketShip = "Falcon Heavy";
    public String poweredBy = "Electricity";
    public boolean inSpace = true;

    public void printName() {
        System.out.println("Hello, my name is "+name);
    }
    public void printAge() {
        System.out.println("I am "+age+" years old");
    }
    public void printRocket() {
        System.out.println("My rocket is "+rocketShip+" and it is powered by "+poweredBy);
    }
    public boolean isInSpace(){
```

	Test	Expected	Got	

✓	`Astronaut starman = new Astronaut();`
	`starman.printName();`
	`starman.printAge();`
	`starman.printRocket();`
	`System.out.println(starman.isInSpace());`
		`Hello, my name is Starman`
		`I am 1 years old`
		`My rocket is Falcon Heavy and it is powered by Electricity`
		`true`
			`Hello, my name is Starman`
			`I am 1 years old`
			`My rocket is Falcon Heavy and it is powered by Electricity`
			`true`
				✓

✓	`Astronaut starman = new Astronaut();`
	`starman.name = "Alien";`
	`starman.age = 328;`
	`starman.rocketShip = "Flying saucer";`
	`starman.poweredBy = "Magic";`
	`starman.inSpace = false;`
	`starman.printName();`
	`starman.printAge();`
	`starman.printRocket();`
	`System.out.println(starman.isInSpace());`
		`Hello, my name is Alien`
		`I am 328 years old`
		`My rocket is Flying saucer and it is powered by Magic`
		`false`
			`Hello, my name is Alien`
			`I am 328 years old`
			`My rocket is Flying saucer and it is powered by Magic`
			`false`
				✓

Passed all tests! ✓

Question author's solution:

```
public class Astronaut {
        public String name = "Starman";
        public int age = 1;
        public String rocketShip = "Falcon Heavy";
        public String poweredBy = "Electricity";
        public boolean inSpace = true;

        public void printName() {
                System.out.println("Hello, my name is " + name);
        }

        public void printAge() {
                System.out.println("I am " + age + " years old");
        }

        public void printRocket() {
                System.out.println("My rocket is " + rocketShip +
                " and it is powered by " + poweredBy);
        }

        public boolean isInSpace() {
                return inSpace;
        }

}
```

Correct

Marks for this submission: 1.00/1.00. Accounting for previous tries, this gives **0.90/1.00**.

Response history

Step	Time	Action	State	Marks
1	4/03/19, 00:49	Started	Not complete	
2	6/03/19, 00:32	Submit: public class Astronaut { /** * Member variables are normally placed at the top * of the class, right below the class name. */ public String name = "Starman"; public int age = 1; public String rocketShip = "Falcon Heavy"; public String poweredBy = "Electricity"; public boolean inSpace = true; public void printName() { System.out.println("Hello, my name is "+name); } public void printAge() { System.out.println("I am "+age+" years old"); } public void printNRocket() { System.out.println("My rocket is "+rocketShip+" and it is powered by "+poweredBy); } public static void main(String[] args) { // Make two objects Astronaut starman = new Astronaut(); starman.printName(); starman.printAge(); starman.printRocket(); System.out.println(starman.isInSpace()); } }	Incorrect	0.00
3	6/03/19, 00:34	Submit: public class Astronaut { /** * Member variables are normally placed at the top * of the class, right below the class name. */ public String name = "Starman"; public int age = 1; public String rocketShip = "Falcon Heavy"; public String poweredBy = "Electricity"; public boolean inSpace = true; public void printName() { System.out.println("Hello, my name is "+name); } public void printAge() { System.out.println("I am "+age+" years old"); } public void printNRocket() { System.out.println("My rocket is "+rocketShip+" and it is powered by "+poweredBy); } public boolean isInSpace(){ return inSpace; } public static void main(String[] args) { Astronaut starman = new Astronaut(); starman.printName(); starman.printAge(); starman.printRocket(); System.out.println(starman.isInSpace()); } }	Incorrect	0.00
4	6/03/19, 00:35	Submit: public class Astronaut { /** * Member variables are normally placed at the top * of the class, right below the class name. */ public String name = "Starman"; public int age = 1; public String rocketShip = "Falcon Heavy"; public String poweredBy = "Electricity"; public boolean inSpace = true; public void printName() { System.out.println("Hello, my name is "+name); } public void printAge() { System.out.println("I am "+age+" years old"); } public void printRocket() { System.out.println("My rocket is "+rocketShip+" and it is powered by "+poweredBy); } public boolean isInSpace(){ return inSpace; } public static void main(String[] args) { Astronaut starman = new Astronaut(); starman.printName(); starman.printAge(); starman.printRocket(); System.out.println(starman.isInSpace()); } }	Correct	0.90
5	**6/03/19, 01:06**	**Attempt finished submitting:**	**Correct**	**0.90**

Information

Java API

Sometimes you may want to use additional methods from other classes that are a part of the standard Java language. This helps with code reuse, so you don't have to reinvent the wheel for basic functions that do well defined tasks, for example String processing.

But where are all of these classes and method documented? In the Java API (your new friend), see https://docs.oracle.com/en/java/javase/11/ (click on "API Documentation" under "Specifications"; then, clicking on "All Classes" in the top left corner of the API documentation will give you a list of all the classes available to you).

You should become very familiar with these resources, since they will be the first port of call when you are looking for a function, and the best resource available to you in the midterm test.

You search the Java API by specifying the class you wish to see. If you were wanting to manipulate strings, then String would be a good place to start.

Some common classes are: String, Integer, Random, Timer, ArrayList (we will talk more about array lists later in the lecture).

Let us start by familiarising ourselves with the reference documentation for the standard library packages (Java API). We have seen the statement System.out.println("Hello New Zealand!") in the lecture:

- What is the type of "System" in the above statement and what class or package is it in? Hint: You may want to have a look at the Java API.
- What is the type of "System.out" in the above statement and what class or package is it in?
- What method is invoked? What class is it in and what is its return type? How many parameters does it have?

Response history

Step	Time	Action	State
1	4/03/19, 00:49	Started	
2	6/03/19, 00:32	Seen	
3	**6/03/19, 01:06**	**Attempt finished**	

Question **5**

Correct

Mark 1.00 out of 1.00

SpaceStrings Class

Open up the String class in the Java API, and have a look at the methods the String class implements.

You can call any of these methods on an instance of a String object, by having objectname.methodName(); for example:

```
String rocketName = "Heart of Gold";
int nameLength = rocketName.length(); // returns 13
```

Also, note that Java supports "escape sequences" in strings. To use escape sequences, we use a backslash ('\') to indicate that the next character is different. For example, to add a tab to a string, we can use "\t", a newline would be "\n", double quotes would be "\"", a single quote would be "\'" and to use a backslash as part of a string we would write "\\".

Your task is to make a new Java class called SpaceStrings that implements the following methods. Note, you will need to look up methods in the Java API to complete this exercise (again, remember to sketch a UML class diagram before coding this class).

- int getStringLength(String str), a method that returns the length of a String.
- int getLetterPosition(String str, char letter), a method that returns the index of the letter in the str
- String makeAllCaps(String str), a method that turns a given str to all caps.
- String makeAllLower(String str), a method that turns a given str to all lowercase.
- String makeHugeString(String str1, String str2), a method that appends str2 to the end of str1.

For example:

Test	Result

`SpaceStrings space = new SpaceStrings();`
 	
`String starman = "Starman Sez";`
`String rocketship = "Red Tesla";`
 	
`System.out.println(space.getStringLength(starman));`
`System.out.println(space.getLetterPosition(starman, 'r'));`
`System.out.println(space.makeAllCaps(starman));`
`System.out.println(space.makeAllLower(starman));`
`System.out.println(space.makeHugeString(starman, rocketship));`
	`11`
	`3`
	`STARMAN SEZ`
	`starman sez`
	`Starman SezRed Tesla`

Answer: (penalty regime: 0, 10, 20, ... %)

```
public class SpaceStrings {
    public int getStringLength(String initString){
        return initString.length();
    }
    public int getLetterPosition(String initString, char ch){
        return initString.indexOf(ch);
    }
    public String makeAllCaps(String initString){
        return initString.toUpperCase();
    }
    public String makeAllLower(String initString){
        return initString.toLowerCase();
    }
    public String makeHugeString(String initString,String str2){
        return initString+str2;
    }

    public static void main(String[] args) {
        SpaceStrings space = new SpaceStrings();

        String starman = "Starman Sez";
```

	Test	Expected	Got	

✓	`SpaceStrings space = new SpaceStrings();`
	`\t\t`
	`String starman = "Starman Sez";`
	`String rocketship = "Red Tesla";`
	
	`System.out.println(space.getStringLength(starman));`
	`System.out.println(space.getLetterPosition(starman, 'r'));`
	`System.out.println(space.makeAllCaps(starman));`
	`System.out.println(space.makeAllLower(starman));`
	`System.out.println(space.makeHugeString(starman, rocketship));`
		`11`
		`3`
		`STARMAN SEZ`
		`starman sez`
		`Starman SezRed Tesla`
			`11`
			`3`
			`STARMAN SEZ`
			`starman sez`
			`Starman SezRed Tesla`
				✓

	Test	Expected	Got	

✓	`SpaceStrings space = new SpaceStrings();`
	
	`String starman = "An Aw3some C00L DuDe";`
	`String rocketship = "HEART of G0Ld";`
	
	`System.out.println(space.getStringLength(starman));`
	`System.out.println(space.getLetterPosition(starman, 'r'));`
	`System.out.println(space.makeAllCaps(starman));`
	`System.out.println(space.makeAllLower(starman));`
	`System.out.println(space.makeHugeString(starman, rocketship));`
		`20`
		`-1`
		`AN AW3SOME C00L DUDE`
		`an aw3some c00l dude`
		`An Aw3some C00L DuDeHEART of G0Ld`
			`20`
			`-1`
			`AN AW3SOME C00L DUDE`
			`an aw3some c00l dude`
			`An Aw3some C00L DuDeHEART of G0Ld`
				✓

Passed all tests! ✓

Question author's solution:

```
public class SpaceStrings {

        public int getStringLength(String str) {
                return str.length();
        }

        public int getLetterPosition(String str, char letter) {
                return str.indexOf(letter);
        }

        public String makeAllCaps(String str) {
                return str.toUpperCase();
        }

        public String makeAllLower(String str) {
                return str.toLowerCase();
        }

        public String makeHugeString(String str1, String str2) {
                return str1.concat(str2);
        }

        public static void main(String[] args) {
                SpaceStrings space = new SpaceStrings();

                String starman = "Starman Sez";
                String rocketship = "Red Tesla";

                System.out.println(space.getStringLength(starman));
                System.out.println(space.getLetterPosition(starman, 'r'));
                System.out.println(space.makeAllCaps(starman));
                System.out.println(space.makeAllLower(starman));
                System.out.println(space.makeHugeString(starman, rocketship));
        }
}
```

Correct

Marks for this submission: 1.00/1.00.

Response history

Step	Time	Action	State	Marks
1	4/03/19, 00:49	Started	Not complete	
2	6/03/19, 00:48	Submit: public class SpaceStrings { public int getStringLength(String initString){ return initString.length(); } public int getLetterPosition(String initString, char ch){ return initString.indexOf(ch); } public String makeAllCaps(String initString){ return initString.toUpperCase(); } public String makeAllLower(String initString){ return initString.toLowerCase(); } public String makeHugeString(String initString,String str2){ return initString+str2; } public static void main(String[] args) { SpaceStrings space = new SpaceStrings(); String starman = "Starman Sez"; String rocketship = "Red Tesla"; System.out.println(space.getStringLength(starman)); System.out.println(space.getLetterPosition(starman, 'r')); System.out.println(space.makeAllCaps(starman)); System.out.println(space.makeAllLower(starman)); System.out.println(space.makeHugeString(starman, rocketship)); } }	Correct	1.00
3	**6/03/19, 01:06**	**Attempt finished submitting:**	**Correct**	**1.00**

Information

Javadoc

The Java API is a very useful tool right? Did you know that Java documentation is generated automatically from Java source code? This means that you too can have nice looking documentation for your Java projects, where Java does all the hard work for you.

Re-use is a core concept in software engineering, and good documentation is important to aiding re-use. One of the prerequisites for successful re-use is the ability to find and evaluate candidates for re-use. In Java, these candidates are likely to be classes, interfaces or methods. The tool javadoc is used to provide information for developers (including our future selves) who need to know more about available services and how to use them.

Lets look into how we can do this ourselves.

javadoc is a command line tool which reads in Java source code, and generates nice HTML documentation by looking at method and variable names, parameters and types. If you write a javadoc comment above a method or variable, it will be included in the generated javadoc.

Lets look at an example.

```java
/**
 * This class implements an Alien that rids on comets through the galaxy
 * These aliens are hardy and dangerous, since they look so ugly you
 * will melt upon seeing them.
 *
 * @author Matthew Ruffell
 * @version 1.1, Feb 2018.
 */
public class CometAlien {

        /**
         * The name of the Alien
         */
        public String name;
        /**
         * The age of the Alien
         */
        public int age;
        /**
         * The height of the Alien
         */
        public int height;
        /**
         * The home place of the Alien
         */
        public String homeGalaxy;

        /**
         * Makes the Alien take grip onto the Comet.
         * Once the Alien holds on, it will never let go
         */
        public void holdOntoComet() {
                // implementation
        }

        /**
         * The Alien will eat Comet dust to survive. Makes the Alien happy.
         *
         * @param comet         The comet to eat dust from
         */
        public void eatCometDust(String comet) {
                // implementation
        }

        /**
         * Returns the Galaxy the comet being ridden is traveling to.
         *
         * @param comet         The comet being ridden.
         * @return                      The galaxy where the comet is going.
         */
        public String getDestination(String comet) {
                // implementation
        }

}
```

We start a Javadoc comment with a standard Java multi line comment of /*, but we add an extra * to make /**. This lets Javadoc know to include it in the documentation, and it still acts like a normal comment. Great scott!

We have comments before the class, variables and methods. To see the output it generates, copy the source code from above and save it as CometAlien.java in a NEW FOLDER, called Doc.

Then in your terminal type -

```
cd Doc
javadoc CometAlien.java
```

To see your Javadoc, open index.html in a web browser.

The last thing to know about are *tags*. Tags let javadoc know of special things like classes, authors, version numbers, method parameters and return objects.

See if you can link the above tags to their outputs in the generated javadoc HTML.

Response history

Step	Time	Action	State
1	4/03/19, 00:49	Started	
2	6/03/19, 00:32	Seen	
3	**6/03/19, 01:06**	**Attempt finished**	

Information

In comes Eclipse ...

Now we have covered some basics its time to introduce the main software we are going to be using this year, and that is Eclipse. Eclipse is a widely used IDE (Integrated Development Environment) for many programming languages, and is frequently selected to be the base of large corporations toolchains, especially in the embedded system world. An IDE is a tool that combines a number of components to create a cohesive environment for the development of software. Some common functions are:

- Text (source code) editor
- Debugging (e.g., to find and fix errors in the programme)
- Versioning system / source code management (e.g., to handle multiple source files created and modified by multiple developers)
- Build tools (e.g., to compile, package, document projects)

In this course we will be using Eclipse for Java. Eclipse is already installed on lab machines, so go ahead and open it up.

Note for own machine users: You can install Eclipse on your own machine too. But first you need to install Java. When you download Java, make sure you get the Java JDK. There are two releases of Java, the JRE and JDK. JRE is a runtime environment for times you don't need to write any code, such as sitting down to build a spaceship in Minecraft. The JDK features Java header files and compilers, which are needed to build your code.

Response history

Step	Time	Action	State
1	4/03/19, 00:49	Started	
2	6/03/19, 00:54	Seen	
3	**6/03/19, 01:06**	**Attempt finished**	

Information

Working with Eclipse

Now that you have installed Eclipse and have launched it, lets run through how it works.

When you first open Eclipse, it will ask you to create a *workspace*. A workspace Is where all of your *projects* will be kept. A project is exactly that, a small project dedicated to one thing, like a Lab or an Assignment. Place the workspace in the default place. After what seems an age, Eclipse will load and present its main window to you.

The Eclipse environment usually has several windows open. These can be rearranged to suit your preferences. Eclipse has the concepts of perspectives to provide sets of appropriate tools for specific tasks or kinds of projects. A perspective is a pre-set layout of views and tool bars. You can see your current perspective at the right end of the menu bar. Note the layout. There is typically an explorer window at the left, a console window at the bottom, perspective-specific other stuff at the right and project files (e.g., your Java source files) in the centre (unless of course you have changed it).

Now that you know what a perspective is look in the file menus for an option to change the perspective to Java. (Window > Open Perspective). Have a look at the new views that show up. It is usual for a language to have its own perspective in Eclipse. It is a fast way to tune the interface for common tasks. If you want to customize a perspective you can either go to Window > Customize Perspective and change everything from the views to the toolbars or you can just add one view what you want via Window > Show View and selecting the desired view.

You can configure some important features of the text editor by selecting Window > Preferences and navigating to "Text Editors" (under "General"). For example, it is highly recommended turn on line numbers.

You could also set your indentation rules. Some companies have strict rules about the number of spaces or tabs you use for indentation in source code. This may seem odd, but having a consistent format is important for readability of your code. It helps avoid version control issues where two nearly identical files show a merge conflict because the whitespace is different.

Now use File > New > Java Project to make your first project.

A wizard will appear and ask you for a name. Name it anything, but be sensible. You don't want the problem of forgetting where your assignment code went, when all your projects are called AA, AAA, AAB. Eclipse may ask you to create a module-info.java file. To keep things simple for now, you can ignore this and should choose to not create that file.

Your project will then be listed in your sidebar. From there you can add new classes by right clicking the src folder, and selecting New > Class.

Note all your code must be placed in the src folder, and if it isn't, Eclipse won't work, and you will end up asking tutors why your code is broken.

Response history

Step	Time	Action	State
1	4/03/19, 00:49	Started	
2	6/03/19, 00:54	Seen	
3	**6/03/19, 01:06**	**Attempt finished**	

Question **6**

Correct

Mark 1.00 out of 1.00

SpaceCalculations

Make a new Java class and call it SpaceCalculations, and save it as SpaceCalculations.java

This class will be used to implement methods that test expressions. Implement the following methods (remember to first sketch a UML class diagram and implementing/testing and running your solution in Eclipse before submitting your answer):

- float addition(float f1, float f2), a method which add two floats together.
- float subtraction(float f1, float f2), a method which subtracts two floats from each other.
- float multiplication(float f1, float f2), a method which multiplies two floats with each other.
- int intDivision(int i1, int i2), a method which performs integer division.
- float floatDivision(float f1, float f2), a method which performs float division.
- double circumference(double radius), a method which computes the circumference from a radius. For reference the formula for circumference is: 2 **π r**
- double toAU(double metres), a method which converts a given length in metres to astronomical units. The conversion units: 149597870700.0 or 6.684587122268445468305e-12 may help, depending on whether you are dividing or multiplying the length. Check the example test cases given.

Note, you will need to use the PI constant from the Math class for circumference. Look it up in Java API. Also, priorities of arithmetic operators in Java are like in most other programming languages, i.e., multiplication/division have higher priority than addition/subtraction, but parentheses can be used to override priority rules. Also, there is a "left to right" association (e.g., 3 * 6 / 2 = 9; 12 / 2 / 3 = 2; 2 * 6 % 2 = 0; 6 % 4 / 2 = 1).

For example:

Test	Result

`SpaceCalculations space = new SpaceCalculations();`
 	
`System.out.println(space.addition(10, 4));`
`System.out.println(space.subtraction(32, 20));`
`System.out.println(space.multiplication(7, 6));`
`System.out.println(space.intDivision(9, 4));`
`System.out.println(space.floatDivision(9,4));`
`System.out.println(space.circumference(8.75));`
`System.out.println(space.toAU(1337133713371337.0));`
	`14.0`
	`12.0`
	`42.0`
	`2`
	`2.25`
	`54.97787143782138`
	`8938.186801153026`

Answer: (penalty regime: 0, 10, 20, ... %)

```java
public class SpaceCalculations {
        public float addition(float fl1, float fl2){
            return fl1+fl2;
        }
        public float subtraction(float fl1, float fl2){
            return fl1-fl2;
        }
        public float multiplication(float fl1, float fl2){
            return fl1*fl2;
        }
        public int intDivision(int fl1, int fl2){
            return fl1/fl2;
        }
        public float floatDivision(float fl1, float fl2){
            return fl1/fl2;
        }
        public double circumference(double db){
            return 2*db*Math.PI;
        }
        public double toAU(double db){
            return db/149597870700.0;
```

	Test	Expected	Got	

✓	`SpaceCalculations space = new SpaceCalculations();`
	
	`System.out.println(space.addition(10, 4));`
	`System.out.println(space.subtraction(32, 20));`
	`System.out.println(space.multiplication(7, 6));`
	`System.out.println(space.intDivision(9, 4));`
	`System.out.println(space.floatDivision(9,4));`
	`System.out.println(space.circumference(8.75));`
	`System.out.println(space.toAU(1337133713371337.0));`
		`14.0`
		`12.0`
		`42.0`
		`2`
		`2.25`
		`54.97787143782138`
		`8938.186801153026`
			`14.0`
			`12.0`
			`42.0`
			`2`
			`2.25`
			`54.97787143782138`
			`8938.186801153026`
				✓

Passed all tests! ✓

Question author's solution:

```
import java.lang.Math;

public class SpaceCalculations {

        public float addition(float f1, float f2) {
                return f1 + f2;
        }

        public float subtraction(float f1, float f2) {
                return f1 - f2;
        }

        public float multiplication(float f1, float f2) {
                return f1 * f2;
        }

        public int intDivision(int i1, int i2) {
                return i1 / i2;
        }

        public float floatDivision(float f1, float f2) {
                return f1 / f2;
        }

        public double circumference(double radius) {
                return 2 * radius * Math.PI;
        }

        public double toAU(double length) {
                return length / 149597870700.0;
        }

        public static void main(String[] args) {
                SpaceCalculations space = new SpaceCalculations();

                System.out.println(space.addition(10, 4));
                System.out.println(space.subtraction(32, 20));
                System.out.println(space.multiplication(7, 6));
                System.out.println(space.intDivision(9, 4));
                System.out.println(space.floatDivision(9,4));
                System.out.println(space.circumference(8.75));
                System.out.println(space.toAU(1337133713371337.0));
        }
}
```

Correct

Marks for this submission: 1.00/1.00.

Response history

Step	Time	Action	State	Marks
1	4/03/19, 00:49	Started	Not complete	
2	6/03/19, 01:06	Submit: public class SpaceCalculations { public float addition(float fl1, float fl2){ return fl1+fl2; } public float subtraction(float fl1, float fl2){ return fl1-fl2; } public float multiplication(float fl1, float fl2){ return fl1*fl2; } public int intDivision(int fl1, int fl2){ return fl1/fl2; } public float floatDivision(float fl1, float fl2){ return fl1/fl2; } public double circumference(double db){ return 2*db*Math.PI; } public double toAU(double db){ return db/149597870700.0; } public static void main(String[] args) { SpaceCalculations space = new SpaceCalculations(); System.out.println(space.addition(10, 4)); System.out.println(space.subtraction(32, 20)); System.out.println(space.multiplication(7, 6)); System.out.println(space.intDivision(9, 4)); System.out.println(space.floatDivision(9,4)); System.out.println(space.circumference(8.75)); System.out.println(space.toAU(1337133713371337.0)); } }	Correct	1.00
3	**6/03/19, 01:06**	**Attempt finished submitting:**	**Correct**	**1.00**

Information

JAR Files

Normally when someone gives you a Java program, it is not in the source code or class file form. Instead, Java programs are commonly distributed as JAR files, or Java ARchives. Essentially they are fancy ZIP archives that hold .class files and have a manifest that tells you what .class holds the main() method. A lot like Android .apk if you have ever looked at them.

Lets make a JAR.

Take your HelloWorld.class file, and make a jar by running

```
jar cfe <jarname> <main class> <class files...>
```

'c' means a new archive is created. 'f' means the result is output to a file, and 'e' means select a main method instead of giving an explicit manifest. You should have something like:

```
jar cfe HelloWorld.jar HelloWorld HelloWorld.class
```

To run your jar, type:

```
java -jar HelloWorld.jar
```

And you should see your result printed in the terminal.

Response history

Step	Time	Action	State
1	4/03/19, 00:49	Started	
2	6/03/19, 00:54	Seen	
3	**6/03/19, 01:06**	**Attempt finished**	

Question **7**

Correct

Mark 1.00 out of 1.00

Running JAR Files

Download the JAR here, and run it. What does it output?

Answer: Whats up, SENG201? ✓

The correct answer is: Whats up, SENG201?

Correct

Marks for this submission: 1.00/1.00.

Response history

Step	Time	Action	State	Marks
1	4/03/19, 00:49	Started	Not complete	
2	6/03/19, 00:54	Saved: Whats up, SENG201?	Not complete	
3	6/03/19, 00:55	Submit: Whats up, SENG201?	Correct	1.00
4	**6/03/19, 01:06**	**Attempt finished**	**Correct**	**1.00**

◄ Quiz 1: Requirements, Stakeholders and Ethics Jump to... JAR Example ►