In this lab, you'll create and access a schedule by using a switch statement. Switch statements provide a structured and efficient way to define, access, and manage schedules in code, promoting clarity, maintainability, and scalability.

Let's use Toni as an example. He has a lot of work to catch up on and keeps forgetting what needs to be done, especially after coming back from his road trip.

Goal

Can you help Toni by creating a program that displays the schedule for the day when the day of the week is given in numeric form? For example, you could represent Monday as 1, Tuesday as 2, and so on.

To help Toni, you'll need to write the necessary code to accomplish the tasks requested. Let's get started!

🖥️ Note: When encountering this icon, it's time to get into your IDE and start coding!

In your lab environment, open IntelliJ by double-clicking on the icon.

When the IntelliJ IDE opens, you will be brought to the starter code.

🖥️ It's time to get coding!

The main method of the main class is placed in a file named **`Main.java`**.

Tip

Always follow best practice when naming files: The name of the class should be the same as the file name.

As soon as you open the Intellij, you will be presented with the following code, in the file named **`Main.java`**.

```
/** TODO 1: Import the Scanner **/
```

```
// Beginning of the main class. The name of the main class and the file should be
same.

public class Main {



    // Beginning of main method - a program starts from this method and ends here

    public static void main(String []args) {



        /** TODO 2: Create a Scanner variable named "keyboard" **/

        Scanner keyboard =



        /** TODO 3: A variable for getting the day of the week is given below **/

        /**         Initialize the variable to have the value 1 **/

        /** 1- is Monday, 2- is Tuesday and so on **/

        int dayOfTheWeek;



        // All other TODOs:



    } // end of main method



} // end of main class
```

The term TODO lets you know your task and the position, within the program, where that task has to be typed in.

The “ALL OTHER TODO’S” is just to show you that there will be different TODO’S like TODO 4, TODO 5 and so on in the actual starter code .

Tip

The term TODO is often used by programmers to mark places where some code needs to be written or changes need to be implemented. It is considered a good practice if you keep a note with TODO in a program to mark the same.

When a TODO is completed, the programmer removes the comment.

If you want toast, you need bread, and if you need bread you have to buy it. Similarly remember, if you want user input, you'll need a **`Scanner`**, and if you need a **`Scanner`**, you'll need to import it! (check out the video *User input for conditional statements* for a refresher)

- 🖥️ TODO 1: Import the **`Scanner`** at the very beginning of the file, before the start of the **`Main`** class, where it says.

```
1
```

```
/** TODO 1: Import the Scanner **/
```

Once you've imported the **`Scanner`**, you'll need to declare it.

- 🖥️ TODO 2: Create a **`Scanner`** variable. Use the hint below. Replace the dots with the code to create the variable, capable of reading from the keyboard. (remember, you need a new one!)

(Confused? Then check out the video *User input for conditional statements* for a refresher)

```
/** TODO 2: Create a Scanner variable named "keyboard" **/

Scanner keyboard = ..................... ;
```

- 🖥️ TODO 3: A variable named **`dayOfTheWeek`** has been declared. Initialize this variable to 1, meaning we are setting it to Monday.

```
/** TODO 3: A variable for getting the day of the week is given below **/

/**         Initialize the variable to have the value 1 **/

/** 1- is Monday, 2- is Tuesday and so on **/

int dayOfTheWeek;
```

How do you know what you can do in a game? How does Toni know what ice cream flavours are there in the shop? You need a Menu or a list of options right? Similarly,

we need to give Toni a list of options by creating a Menu. Let us display the menu first.

Tip

In Java, `/**` is used to start a multi-line comment, and `*/` is used to end it. This type of comment is often used for documentation or when blocking out sections of code. If you want to use the starter code provided and add `println` statements (or any other code) inside a section marked with these comments, you will need to delete both the `/**` at the beginning and the `*/` at the end. Otherwise, your code will still be treated as a comment and won't run.

- 🖥️ TODO 4: Create the menu. You want your program to display the options for the user. (The sample below is one way your menu could look.)

```
/** TODO 4: Display a menu to get the day of the week from the user **/



/********** MENU *****************

These are the choices for week of the day.

Please enter only values from 1-7:

1. MONDAY

2. TUESDAY

3. WEDNESDAY

4. THURSDAY

5. FRIDAY

6. SATURDAY

7. SUNDAY

*********************************/
```

(Don’t forget to run the program and make sure the menu looks the way you want it to!)

Alright, so how will you offer Toni a way to choose the day of the week? You'll need a prompt, right? And then you'll need to read the integer Toni enters into the variable **`dayOfTheWeek`**.

- 🖥️ TODO 5: To complete the following code provided, use the keyboard variable to read an integer into the variable **`dayOfTheWeek`**. Replace the `- - - - - - - -` with the required variable. Don’t forget to complete a prompt.

```
/** TODO 5: Prompt Toni for input. Also let Toni know the valid choices like (1-7).
**/

System.out.println("Enter your choice of the day (1-7):");

dayOfTheWeek = --------.nextInt();
```

You need to check if the user input is equal to any of the values from 1 to 7. Which means we have multiple equality checks. So, which construct is best suited for multiple equality checks? If, else-if, else, or switch?

Definitely a **`switch`** right?

(Want to remind yourself about a **`switch`**, check out the video, The switch statement and Common uses for switch statements.)

Now you will use the **`switch`** statement to help choose Toni’s schedule, for a particular day, based on an equality comparison of the value of the variable **`dayOfTheWeek`**, the value entered by Toni.

- 🖥️ TODO 6: The **`switch`** and the 1st **`case`** is done for you. Implement the rest of the **`case`** statements.

```
/** TODO 6: Create a switch with case values from 1-7 to handle the schedule **/

/**                for the input in "dayOfTheWeek" **/

switch (dayOfTheWeek) {

    case 1:

        System.out.println("Monday: It is back to work.....");

        break;
```

```
    /** TODO 6(a): Implement all other cases for Tuesday to Sunday **/



    /** TODO 6(b): Implement "default" to handle Toni entering **/

    /**            values less than 1 or more than 7 **/

}
```

Well done on keeping up! You have successfully implemented a menu and also how to execute different actions based on user choice.

OK, so far so good. But what will happen if Toni enters a value that is not between 1 and 7, like 9? Run your program and try it!

The values not between 1 and 7 are handled by the **`default`**.

Run the program and input **`dayOfTheWeek`** as 1, when prompted. You will observe that it executes **`case 1`** and then all the rest.

What just happened? Are you flabbergasted? Don't be. Do you remember the fall through we discussed in the video, The switch statement?

- 🖥️ TODO 7: Include a **`break`** statement after the end of the code for every **`case`** statement to prevent fall through. Handle **`Case 1`** as shown below. And repeat for all the other cases.

```
/** TODO 7: Introduce a "break" after each case code, except the last **/
```

```
/** TODO 6: Create a switch with case values from 1-7 to handle the schedule **/

/**                for the input in "dayOfTheWeek" **/

switch (dayOfTheWeek) {

    case 1:

        System.out.println("Monday: It is back to work.....");
```

```
        break;


    /** TODO 6(a): Implement all other cases for Tuesday to Sunday **/


    /** TODO 6(b): Implement "default" to handle Toni entering **/
    /**            values less than 1 or more than 7 **/
}
```

Remember

To follow best practice, always use a **`break`** after each **`case`**, and ensure your **`default`** is the last **`case`** in your **`switch`**.

OK, it's time to test your program. Start by entering 1 and then 2 and so on up to 7 and run each time.

It should run perfectly now. If not, go over the steps carefully to determine if you missed a few **`break`** statements.

To make the program a bit more interactive, imagine Toni running off to work on Monday, but then often forgets to have breakfast on Monday, in which case a diversion is on the books. So shall we ask if Toni had done breakfast and then make a suggestion based on the input Toni makes?

- 🖥️ TODO 8: In order to get input as to whether Toni has had breakfast or not, you need to store the input. So what do you need for that? Yes, you are correct, another variable. That is what the next TODO is about.

```
1
2
3
/** TODO 8: Go below the TODO 2 after the creation of Scanner variable **/
/**            and declare a variable "subChoice" **/
```

Tip

It is a best practice to declare variables at the start of the main method.

Variables are written in camel case, starting with a lower case.

So, Toni is going to have options on Monday, ain't he? How does he know his options? Think you just gave him options before. Yes, you are correct, it is a menu. So, let us create one, inside **`case 1`**, which represents Monday.

- 🖥️ TODO 9: Create a menu to display inside **`case 1`** (you can create your own choices for Toni).

```
/** TODO 9: Move below the display "Monday: It is back to work....." **/

/**            in case 1 and create a menu **/

/** The sub-menu inside case 1 show appear as below:



********** SUB MENU FOR MONDAY ******

* 1. Yes, I had my breakfast

2. No, I would like one

************************************



**/
```

Run your code with a value of 1, did you get the menu you wanted? If yes, congratulations, you are getting better at menu creation. If not, go back and check the steps so far. Now, let's move to the next step.

You have a sub-menu for Monday.

- 🖥️ TODO 10: Now prompt Toni for his choice and on this sub-menu, and get the input.

```
/** TODO 10: After the sub-menu inside case 1, prompt for input on choice **/

/**            from Toni and store it in the variable subChoice **/
```

- 🖥️ TODO 11: Now, why don't you display the advice to Toni, based on his choice of the sub-menu for Monday. You can use an if-else chain for this.

```
/** TODO 11: Display appropriate response to Toni, based on value in **/

/**            variable subChoice using if, if-else, else **/

/**

    1. When subChoice has value of 1 display -

        Good. Now you have a catchup meeting at 9AM Prepare your notes



    2. When subChoice has value of 2 display -

        Lets us divert on Highway 54 and have a Burger on the drive in



    3. When subChoice has any other value then 1 or 2 display -

        NO such choice available

**/
```

Let's say Monday is done. Now would be a good time to expand the program to include the same type of thing for the other days. How about asking Toni a few more questions to enable him to function better on other days?

- 🖥️ TODO 12: If the previous code outputs the basic answers for the other days, why not try to spice it up a bit? How about, on Tuesday, you ask Toni whether the language he'll be using during coding time is C++ or Java? Then on Wednesday, when Toni designs the program, ask if he uses a "Simple Design" or "Complex Design". Once Toni starts coding on Thursday, you could ask him if the IDE is "IntelliJ", "Eclipse" or "VS Code".

Finally, on Friday, when he runs tests, you could ask if there were errors in the tests. And for a bit of fun, ask if Toni was going to chill "at home", "in a bar", "with friends" or "at a nightclub". Then make sure the appropriate responses are displayed.

For `case 2` (`Tuesday`), `case 4` (`Thursday`), and `case 7` (`Sunday`), use a nested `switch` statement, and if-else for the rest. This is because these days have more than one possible outcome depending on another factor.

For instance, on `Tuesday` (`case 2`), Toni might have different tasks depending on the programming language he is using that day. This can be represented as a nested `switch` statement within the case for Tuesday, with different tasks for each programming language.

- 🖥️ TODO 13: For the remaining days (`case 1`, `case 3`, `case 5`, and `case 6`), the activities are straightforward and do not depend on another variable, so a simple if-else statement is used.

Remember

Always test your code. Enter and test all expected possible values and ensure you get the output you were expecting.

Wow, you have successfully completed the tasks. Let's recap what you've done so far—you helped Toni to remember his schedule, based on the weekday Toni inputs. Then you refined the schedule to incorporate further queries to enable Toni to accomplish specialized tasks on the selected weekday, by using nested `switch` statements, nested if statements, and other constructs, which are allowed by Java.