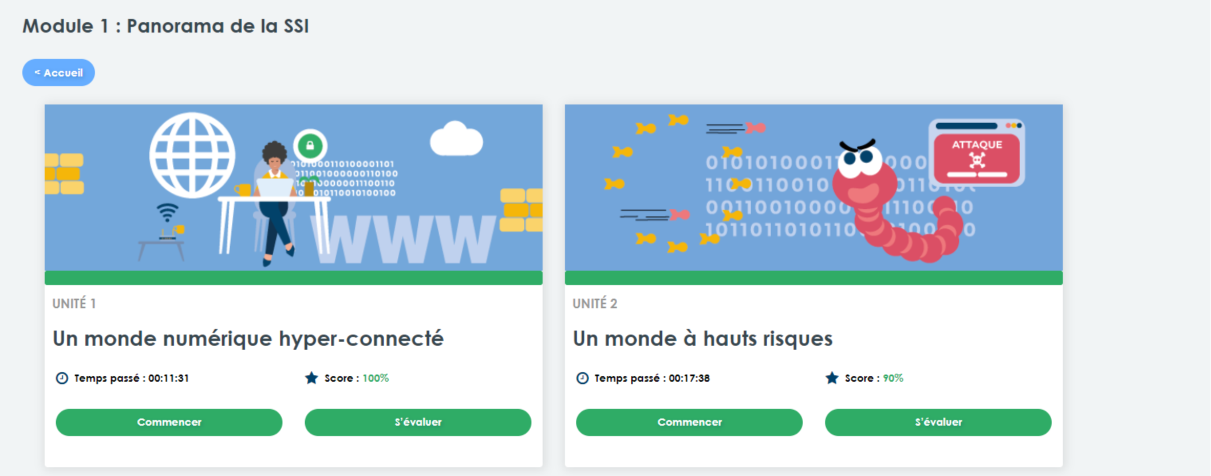
Module 1 :



Module 1 : Panorama de la SSI

UNITÉ 1

Un monde numérique hyper-connecté

Temps passé : 00:11:31

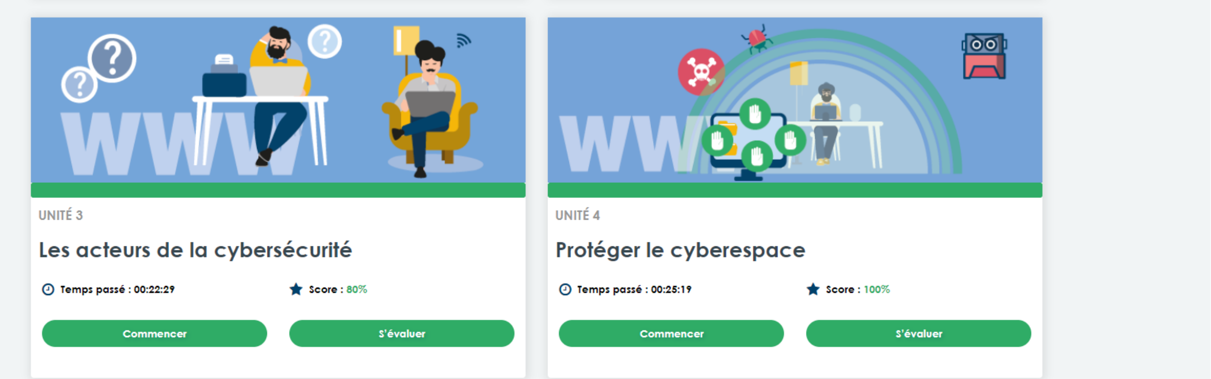
Score : 100%

UNITÉ 2

Un monde à hauts risques

Temps passé : 00:17:38

Score : 90%



UNITÉ 3

Les acteurs de la cybersécurité

Temps passé : 00:22:29

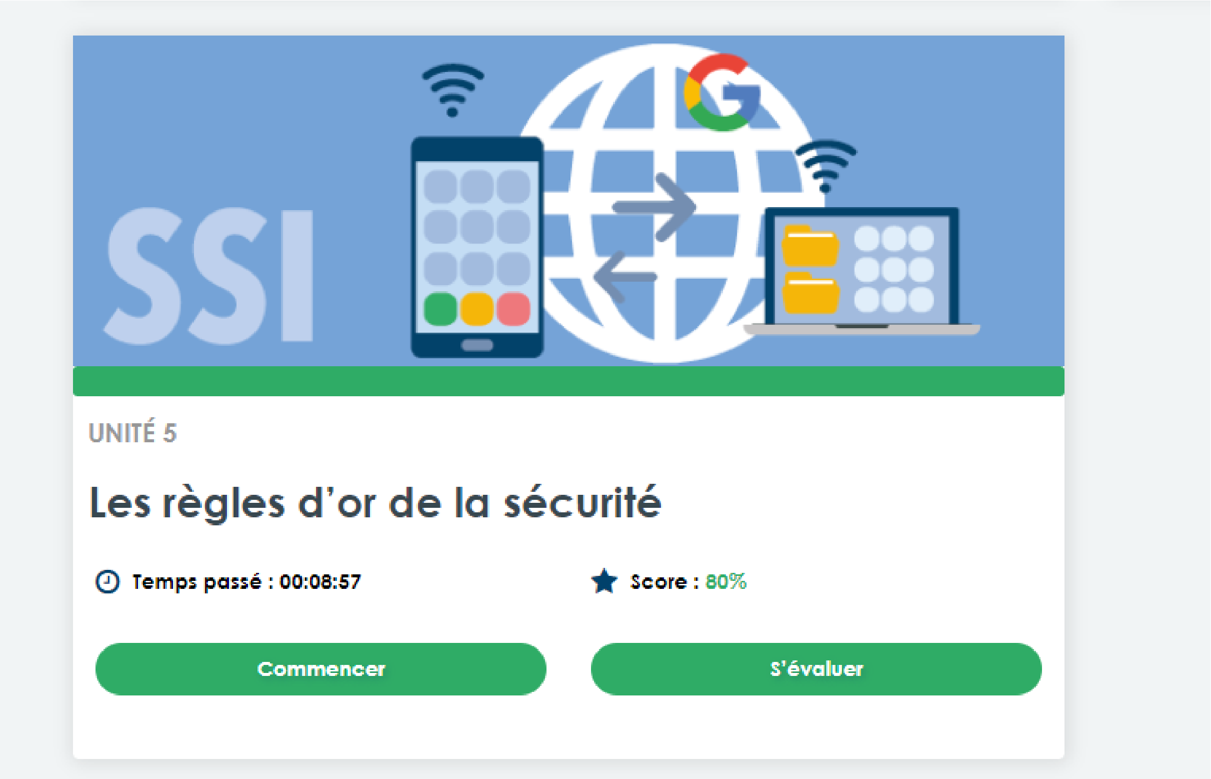
Score : 80%

UNITÉ 4

Protéger le cyberespace

Temps passé : 00:25:19

Score : 100%



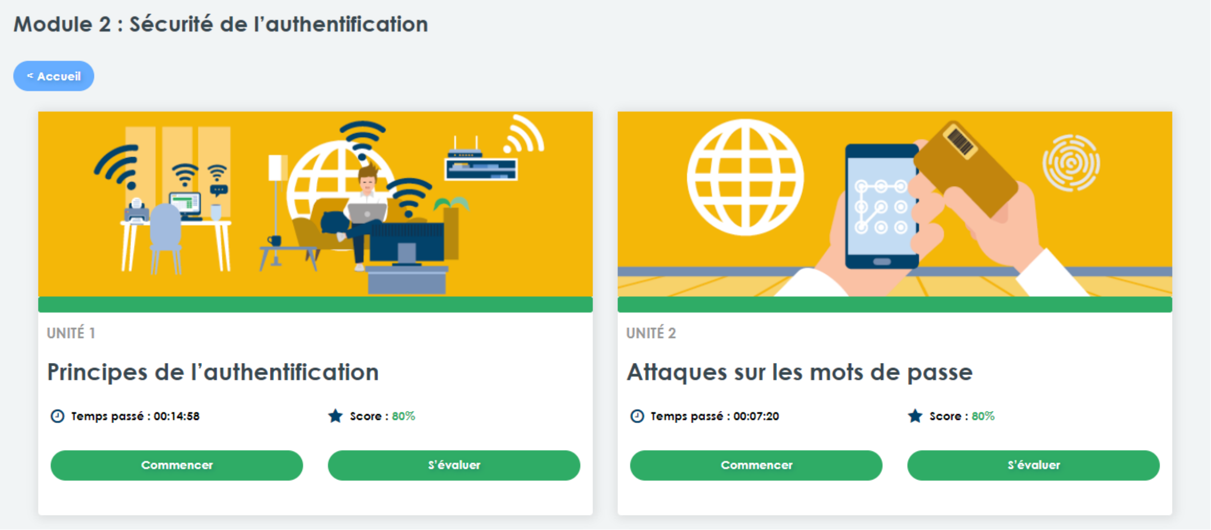
UNITÉ 5

Les règles d'or de la sécurité

Temps passé : 00:08:57

Score : 80%

Module 2 :



Module 2 : Sécurité de l'authentification

UNITÉ 1

Principes de l'authentification

Temps passé : 00:14:58

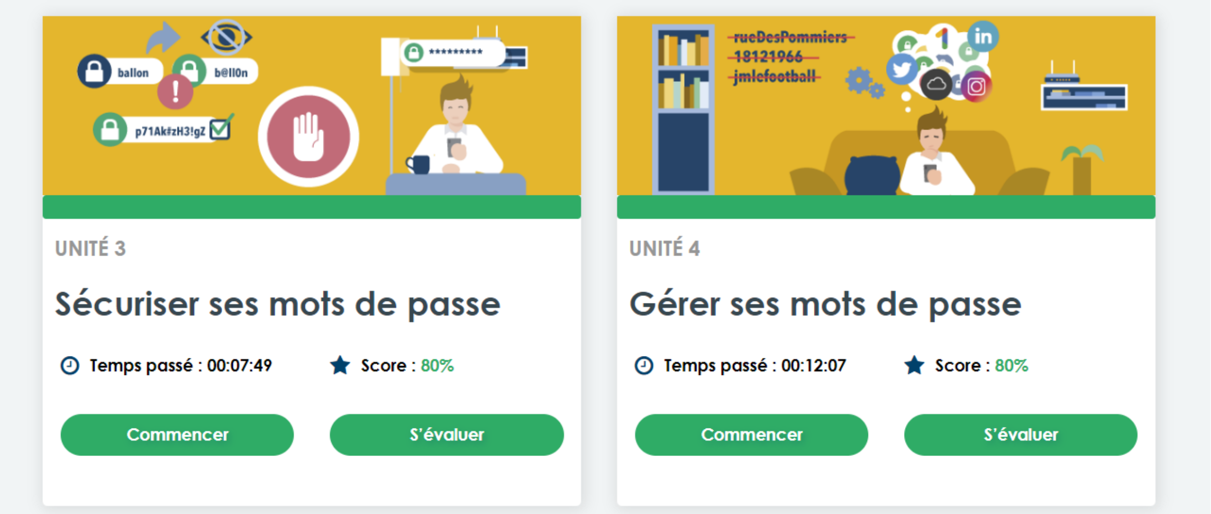
Score : 80%

UNITÉ 2

Attaques sur les mots de passe

Temps passé : 00:07:20

Score : 80%



UNITÉ 3

Sécuriser ses mots de passe

Temps passé : 00:07:49

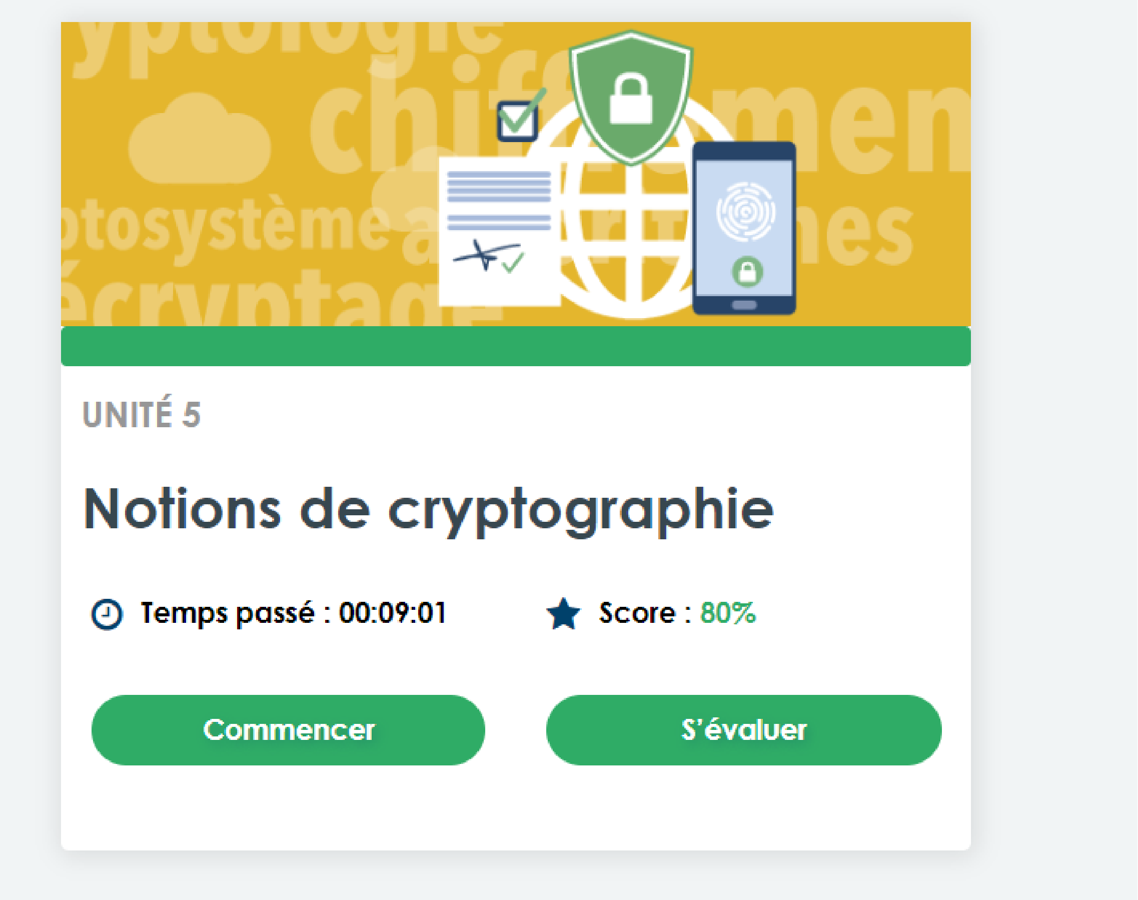
Score : 80%

UNITÉ 4

Gérer ses mots de passe

Temps passé : 00:12:07

Score : 80%

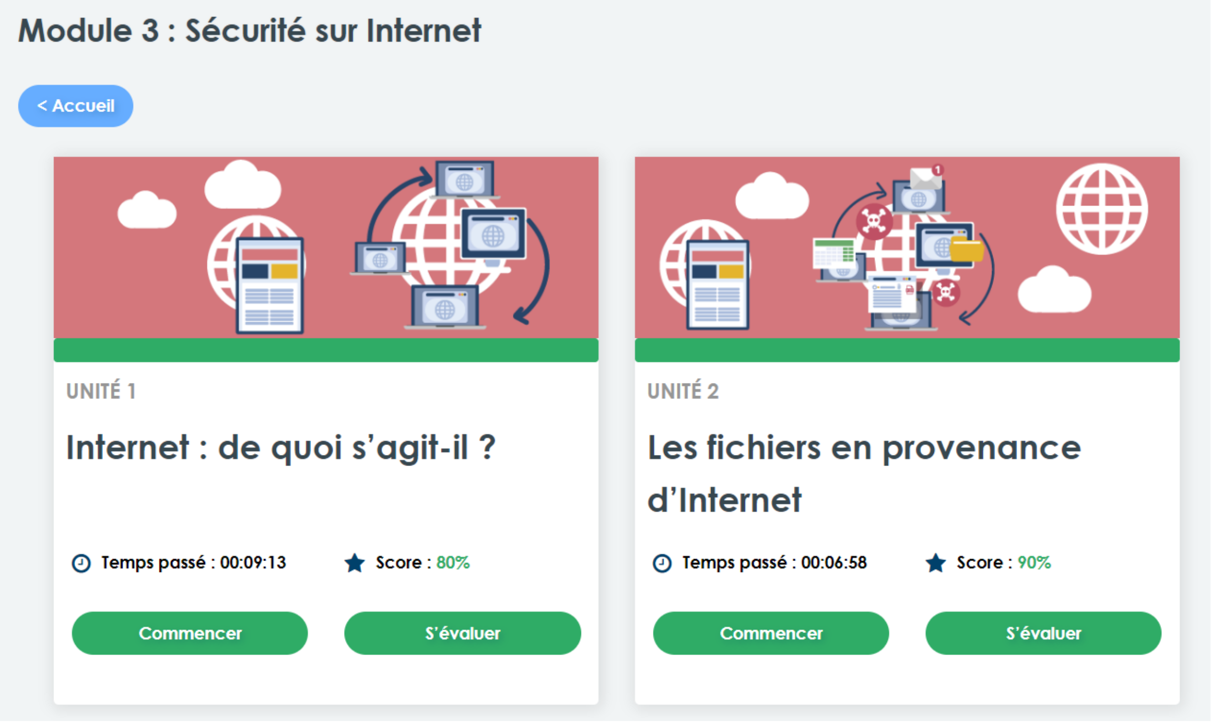


UNITÉ 5

Notions de cryptographie

Temps passé : 00:09:01 Score : 80%

Module 3 :



Module 3 : Sécurité sur Internet

UNITÉ 1

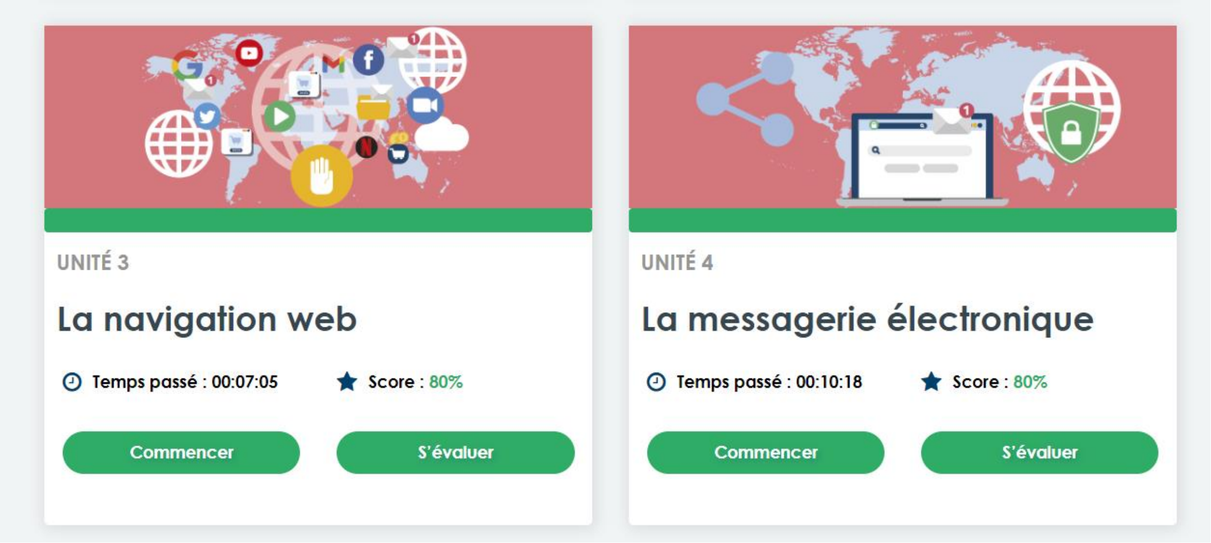
Internet : de quoi s'agit-il ?

Temps passé : 00:09:13 Score : 80%

UNITÉ 2

Les fichiers en provenance d'Internet

Temps passé : 00:06:58 Score : 90%



UNITÉ 3

La navigation web

Temps passé : 00:07:05

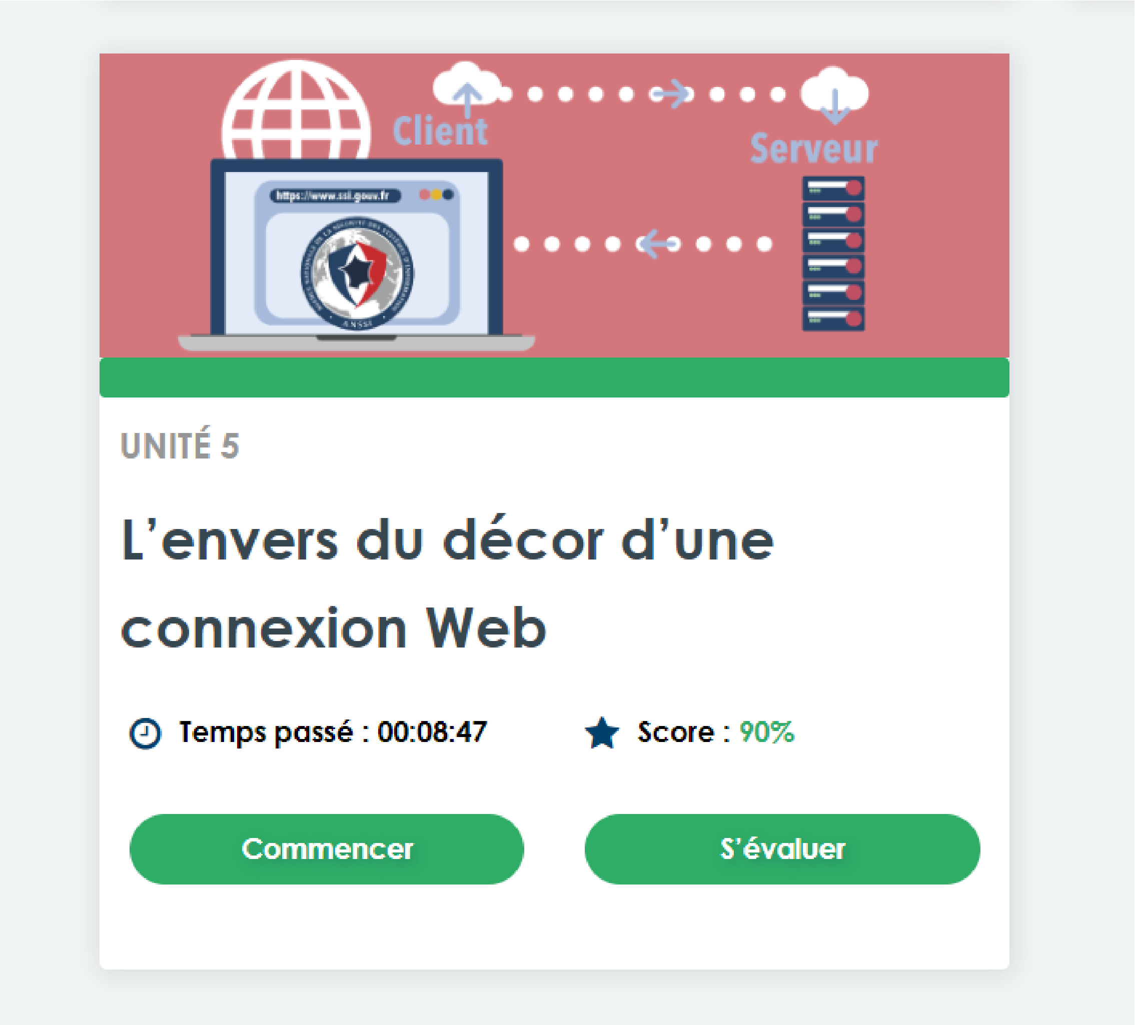
Score : 80%

UNITÉ 4

La messagerie électronique

Temps passé : 00:10:18

Score : 80%



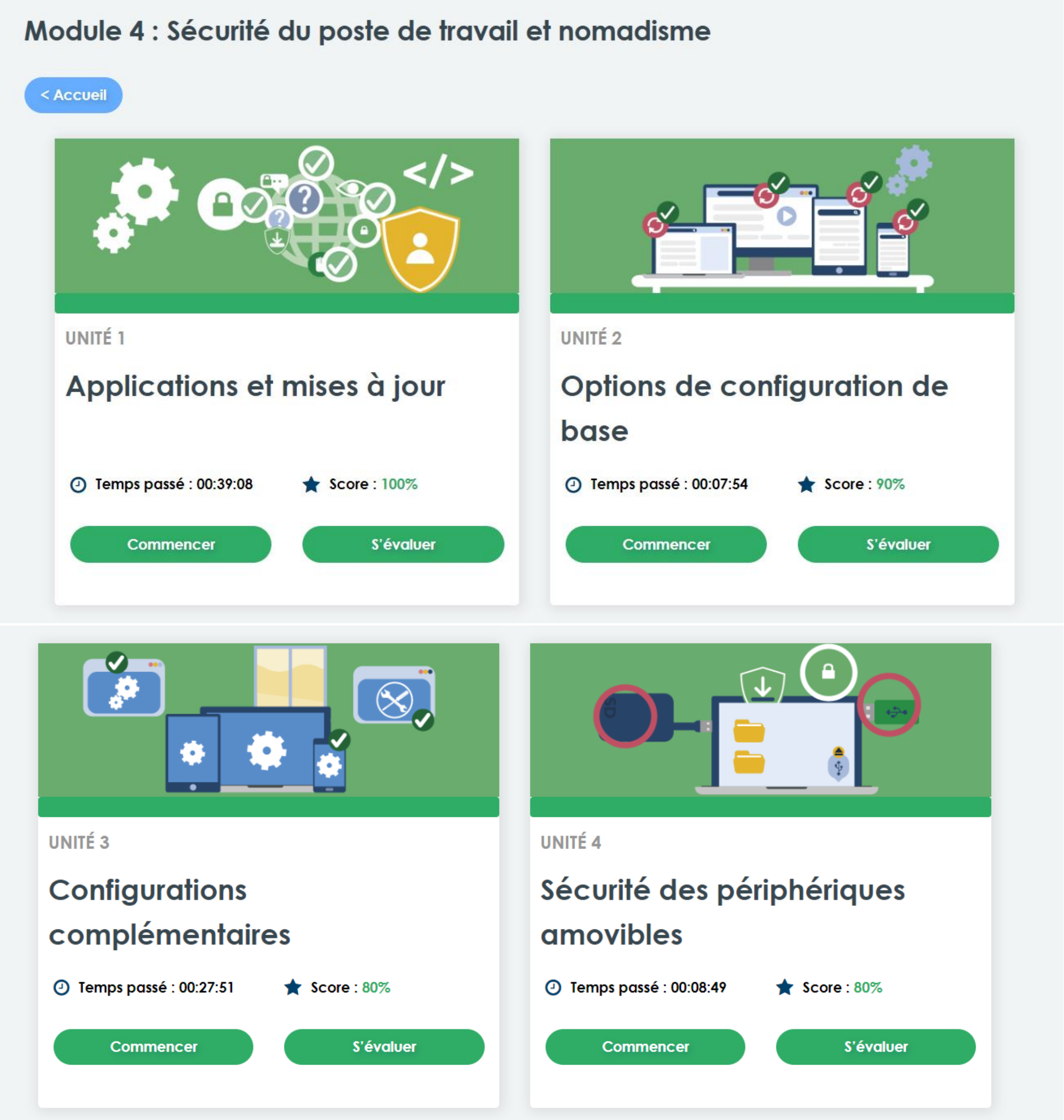
UNITÉ 5

L'envers du décor d'une connexion Web

Temps passé : 00:08:47

Score : 90%

Module 4 :



Module 4 : Sécurité du poste de travail et nomadisme

UNITÉ 1

Applications et mises à jour

Temps passé : 00:39:08 ★ Score : 100%

UNITÉ 2

Options de configuration de base

Temps passé : 00:07:54 ★ Score : 90%

UNITÉ 3

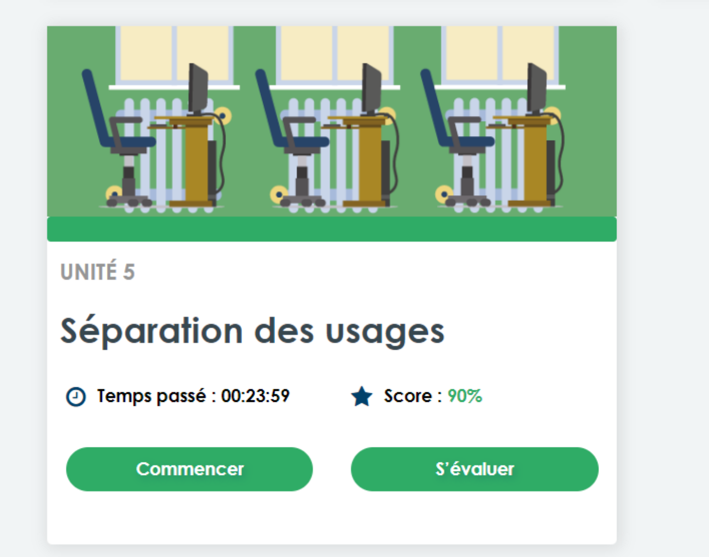
Configurations complémentaires

Temps passé : 00:27:51 ★ Score : 80%

UNITÉ 4

Sécurité des périphériques amovibles

Temps passé : 00:08:49 ★ Score : 80%

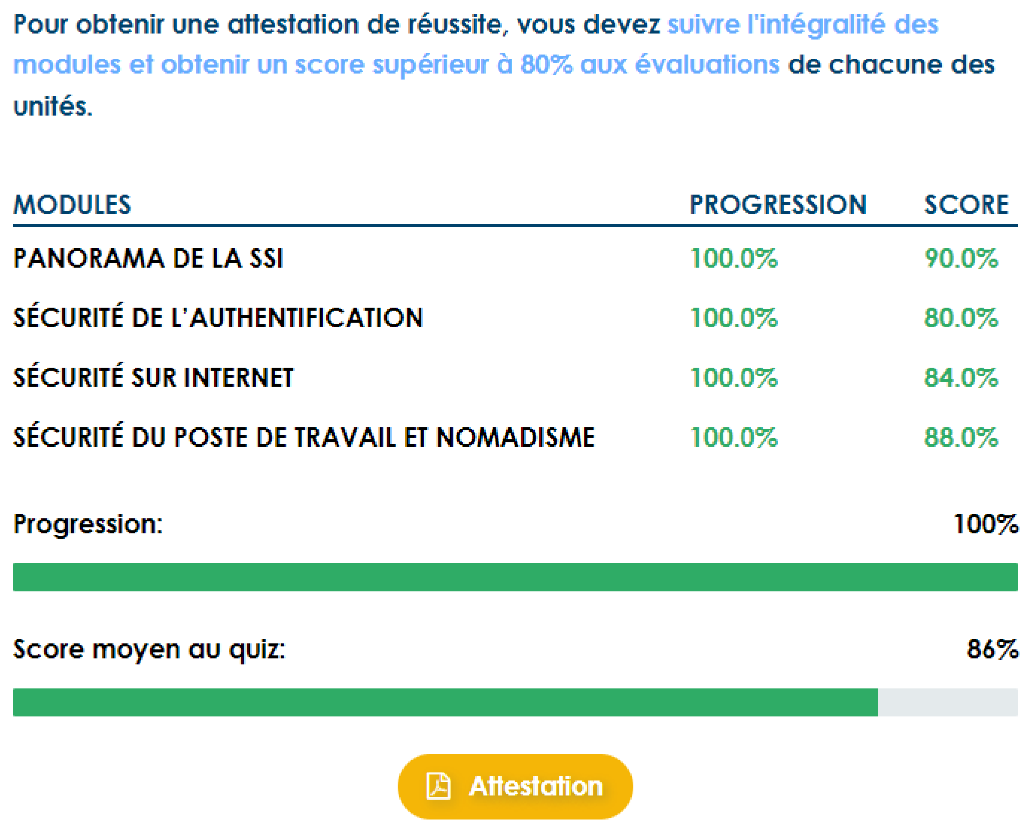


UNITÉ 5

Séparation des usages

Temps passé : 00:23:59 ★ Score : 90%

Preuve attestation :



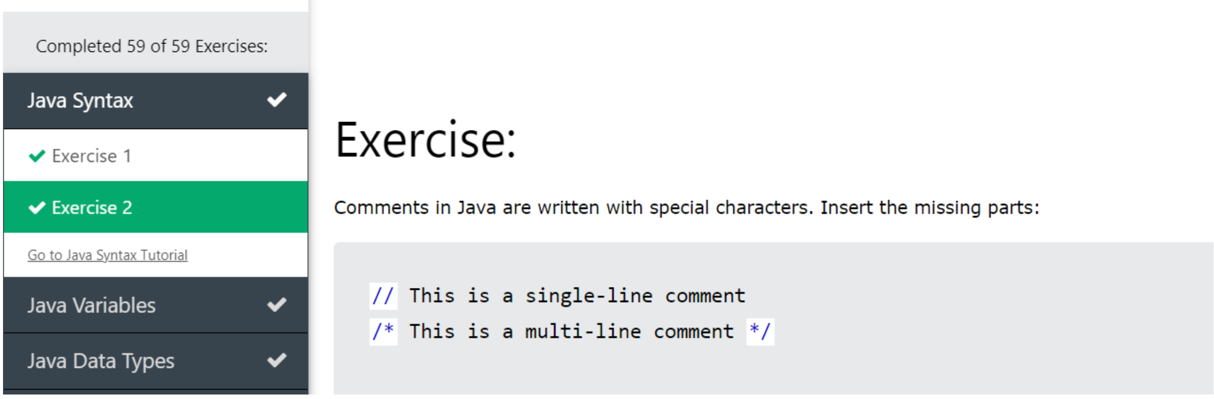
Pour obtenir une attestation de réussite, vous devez suivre l'intégralité des modules et obtenir un score supérieur à 80% aux évaluations de chacune des unités.

MODULES	PROGRESSION	SCORE
PANORAMA DE LA SSI	100.0%	90.0%
SÉCURITÉ DE L'AUTHENTIFICATION	100.0%	80.0%
SÉCURITÉ SUR INTERNET	100.0%	84.0%
SÉCURITÉ DU POSTE DE TRAVAIL ET NOMADISME	100.0%	88.0%

Progression: 100%

Score moyen au quiz: 86%

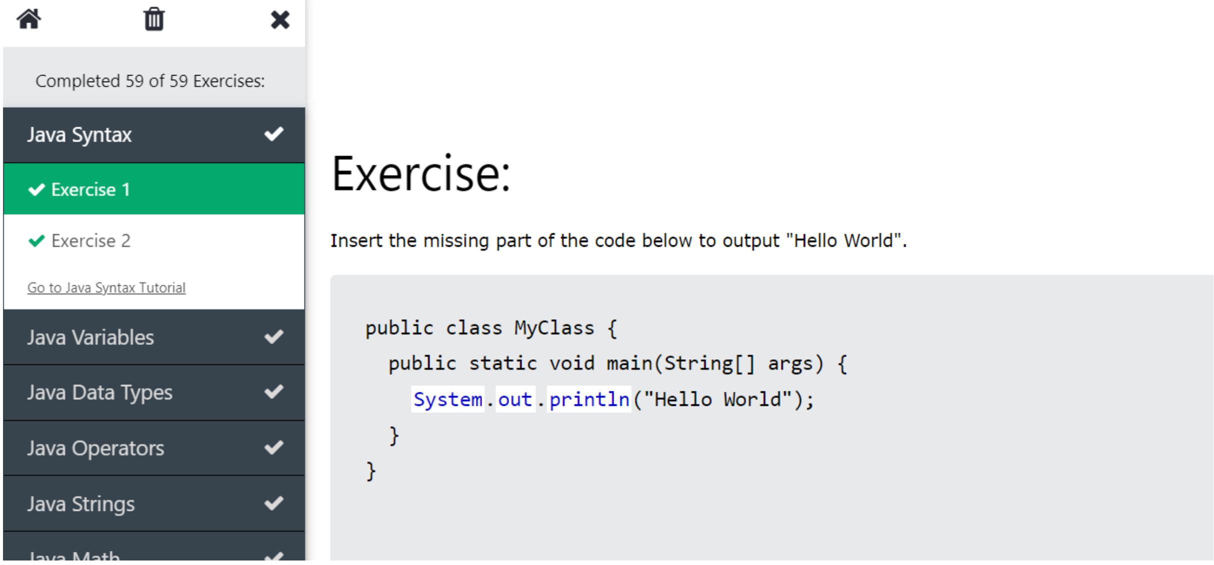
Exo sur le langage Java :



Exercise:

Comments in Java are written with special characters. Insert the missing parts:

```
// This is a single-line comment
/* This is a multi-line comment */
```

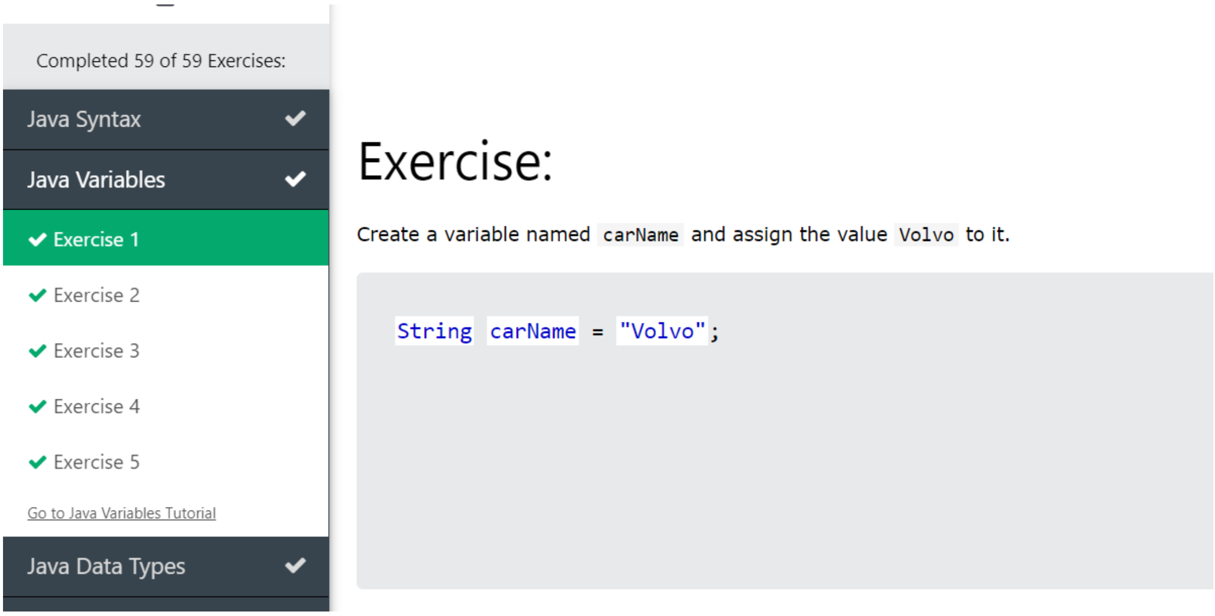


Exercise:

Insert the missing part of the code below to output "Hello World".

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

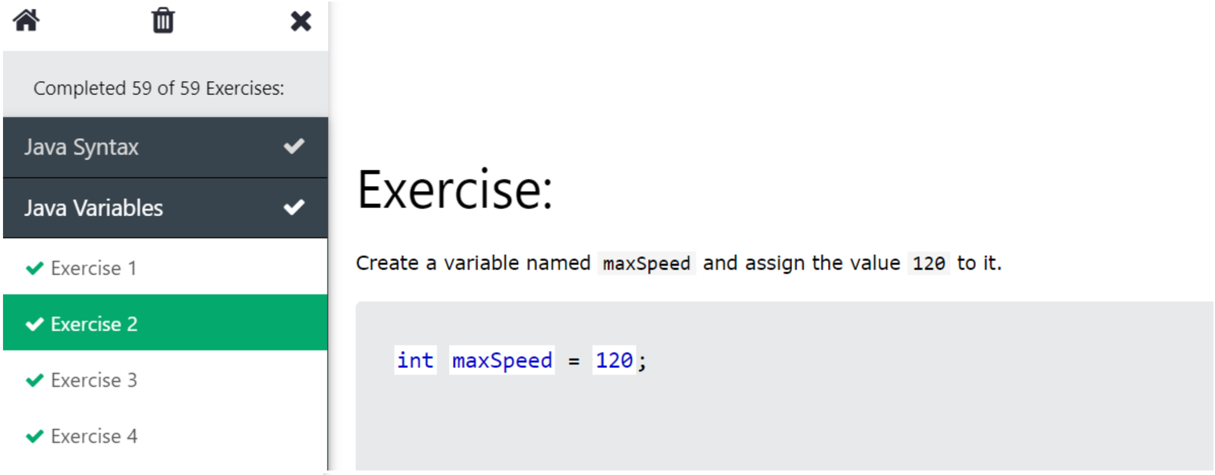
Exo sur les variables :



Exercise:

Create a variable named `carName` and assign the value `Volvo` to it.

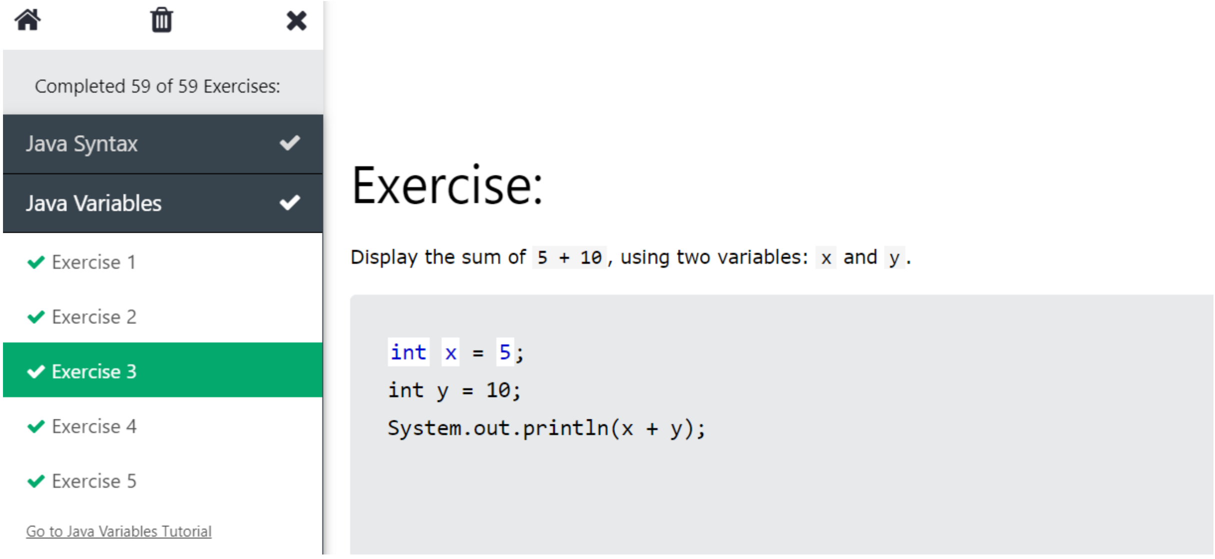
```
String carName = "Volvo";
```



Exercise:

Create a variable named `maxSpeed` and assign the value `120` to it.

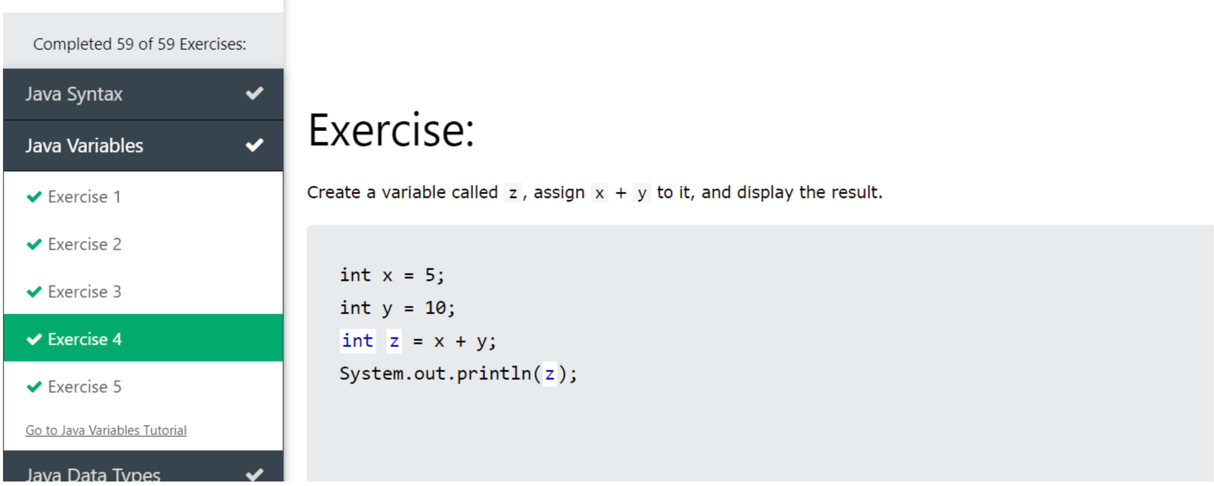
```
int maxSpeed = 120;
```



Exercise:

Display the sum of `5 + 10`, using two variables: `x` and `y`.

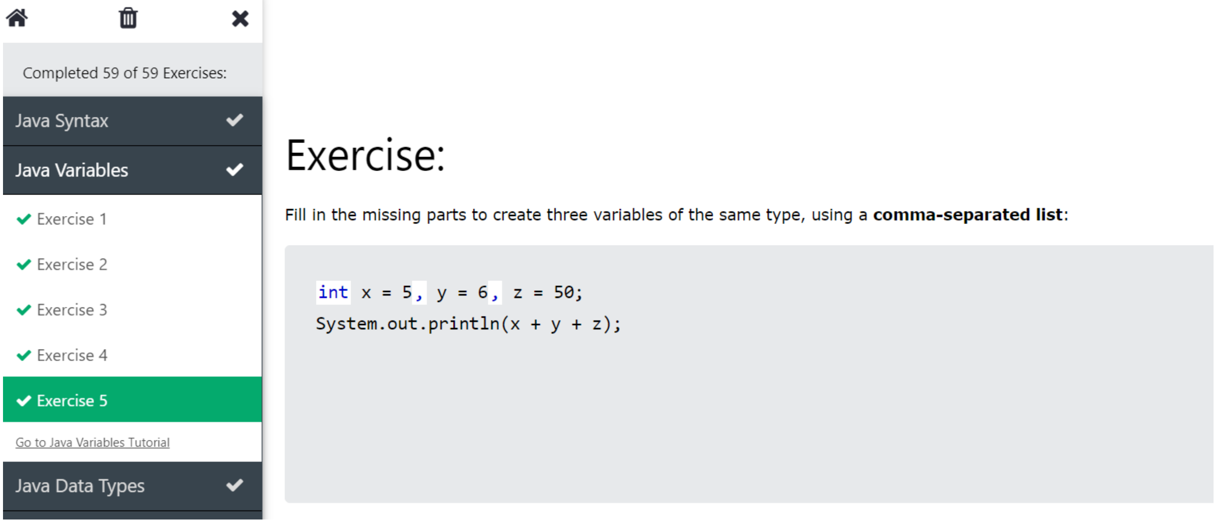
```
int x = 5;
int y = 10;
System.out.println(x + y);
```



Exercise:

Create a variable called `z`, assign `x + y` to it, and display the result.

```
int x = 5;
int y = 10;
int z = x + y;
System.out.println(z);
```

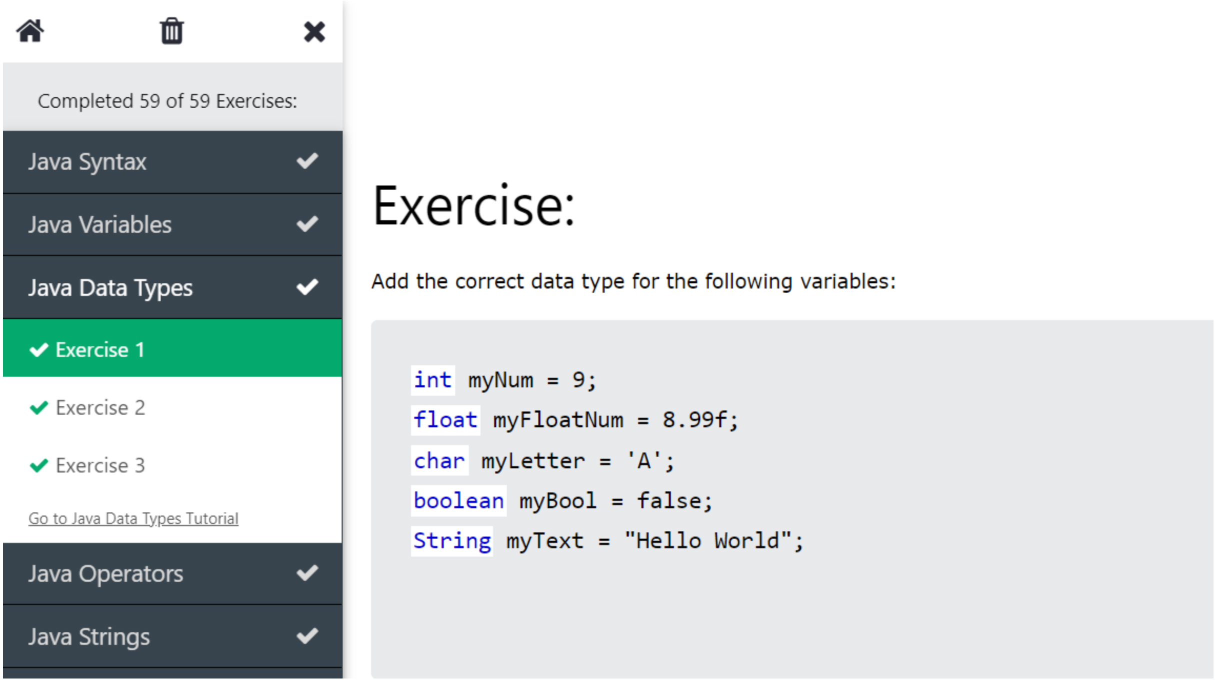


Exercise:

Fill in the missing parts to create three variables of the same type, using a **comma-separated list**:

```
int x = 5, y = 6, z = 50;
System.out.println(x + y + z);
```

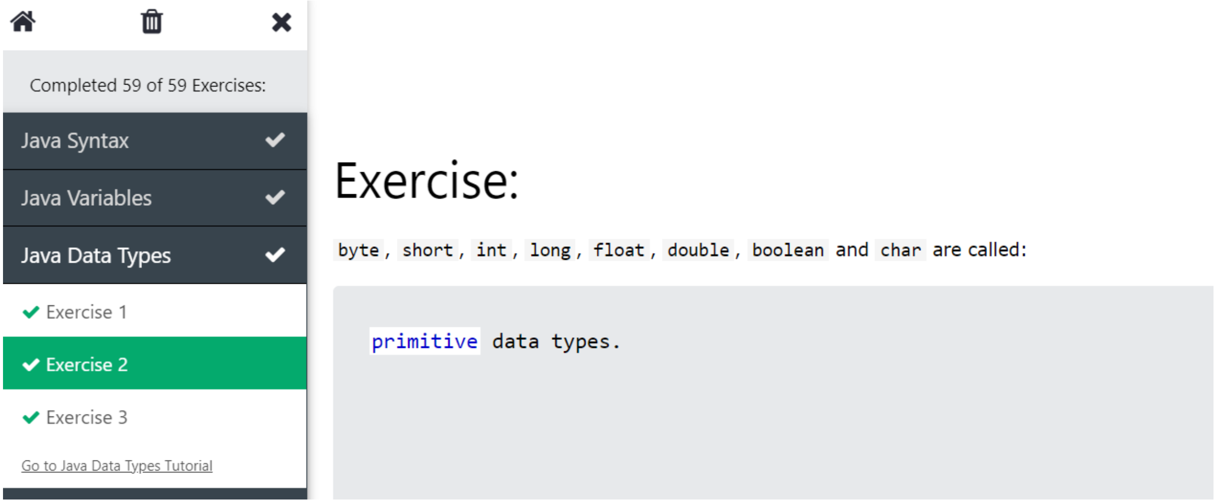
Exo sur les différents types java :



Exercise:

Add the correct data type for the following variables:

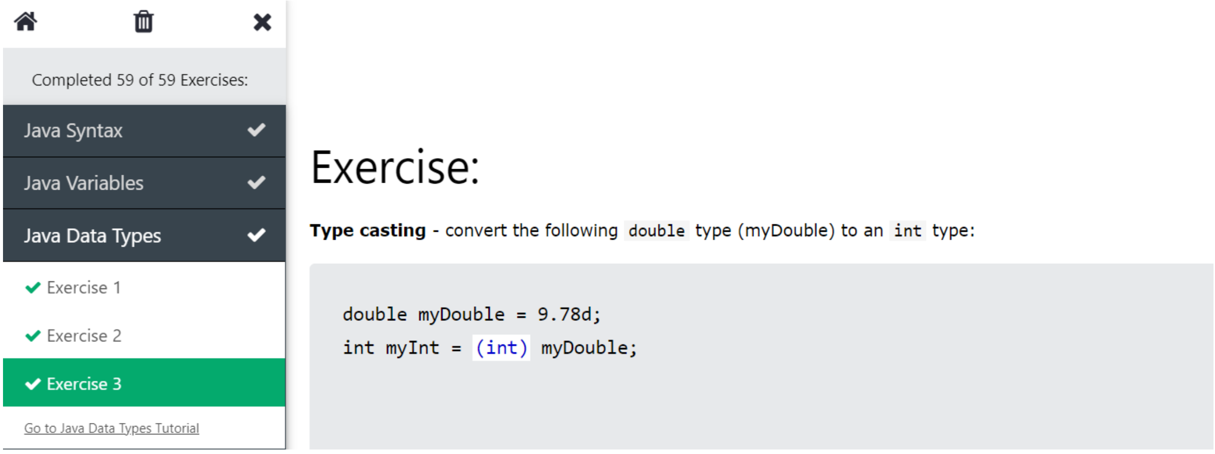
```
int myNum = 9;
float myFloatNum = 8.99f;
char myLetter = 'A';
boolean myBool = false;
String myText = "Hello World";
```



Exercise:

`byte` , `short` , `int` , `long` , `float` , `double` , `boolean` and `char` are called:

```
primitive data types.
```

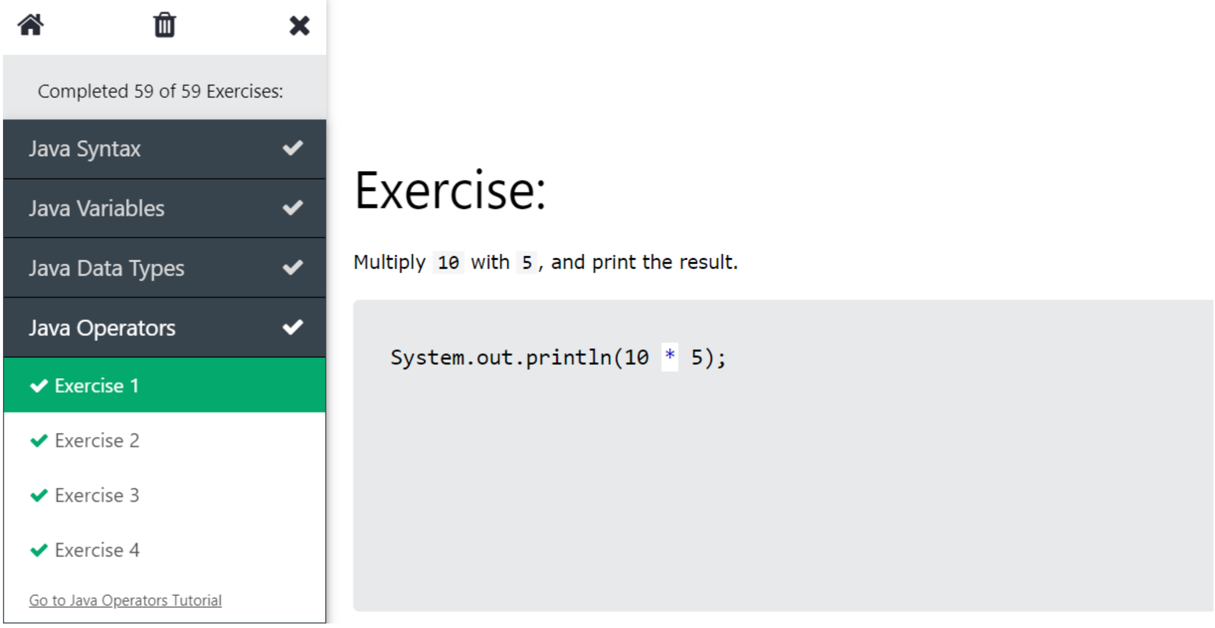


Exercise:

Type casting - convert the following `double` type (myDouble) to an `int` type:

```
double myDouble = 9.78d;
int myInt = (int) myDouble;
```

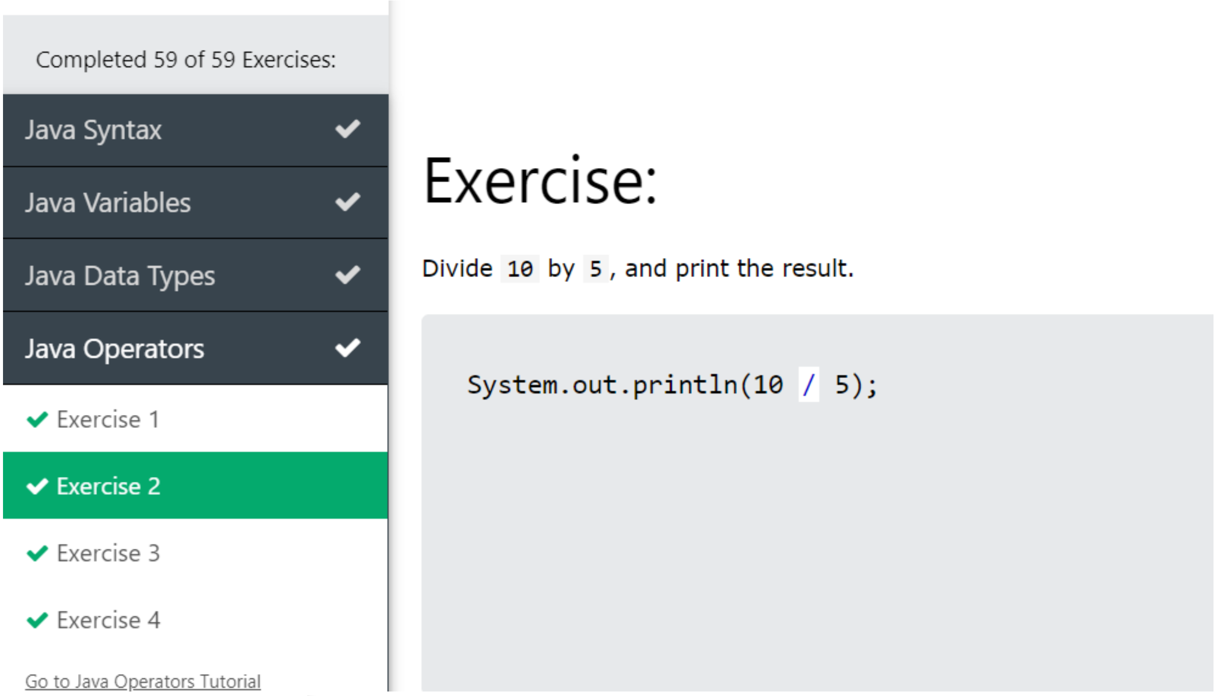
Exo sur les opérations :



Exercise:

Multiply `10` with `5`, and print the result.

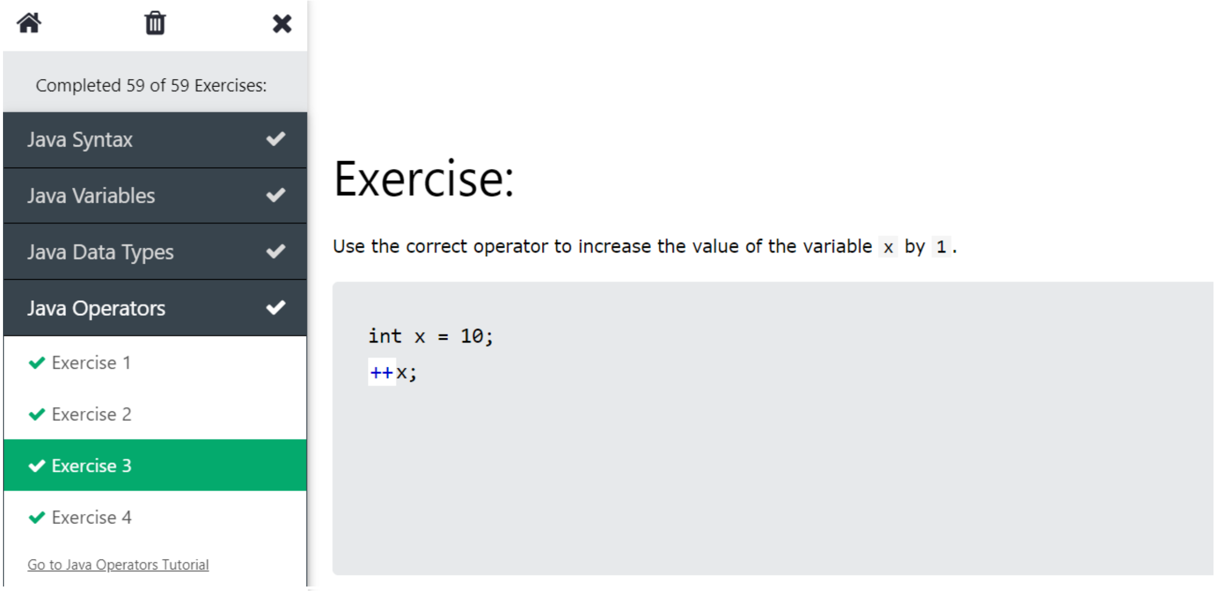
```
System.out.println(10 * 5);
```



Exercise:

Divide `10` by `5`, and print the result.

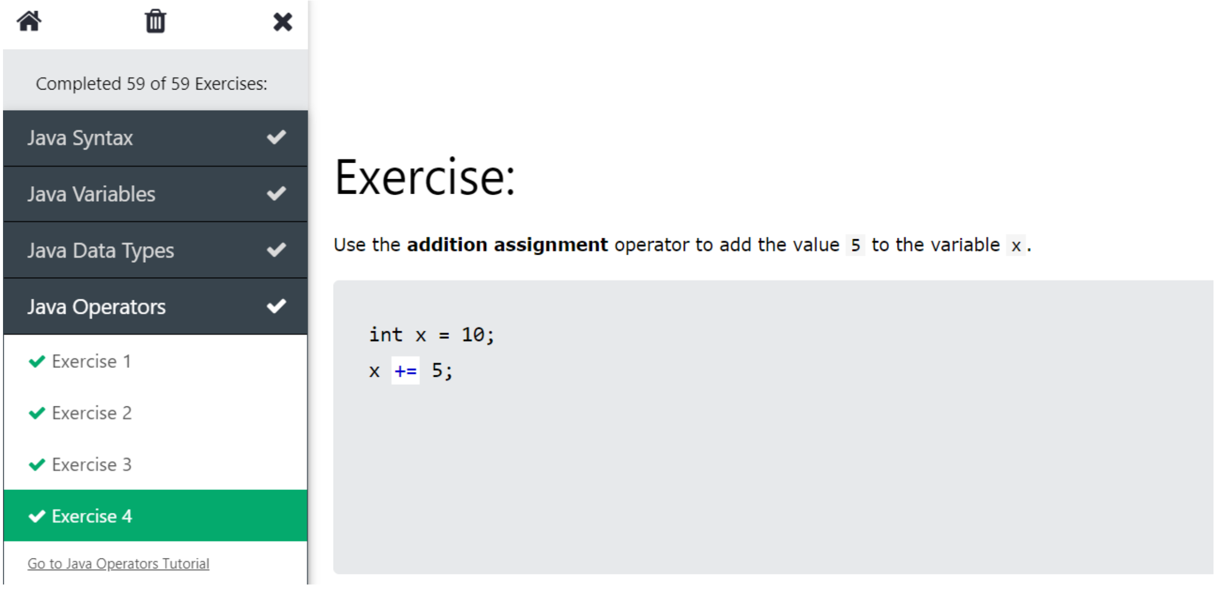
```
System.out.println(10 / 5);
```



Exercise:

Use the correct operator to increase the value of the variable `x` by `1`.

```
int x = 10;
++x;
```

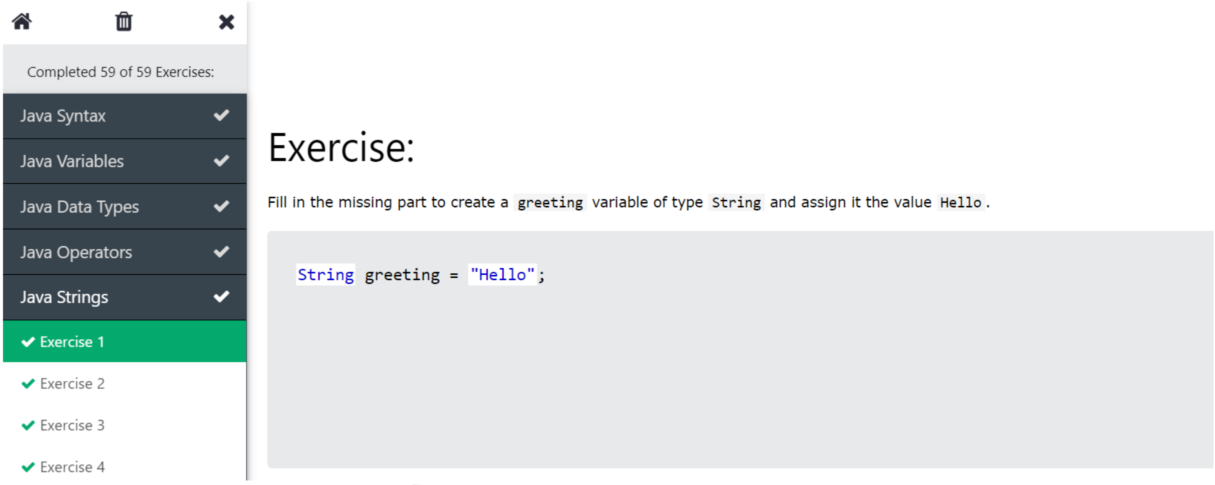


Exercise:

Use the **addition assignment** operator to add the value `5` to the variable `x`.

```
int x = 10;
x += 5;
```

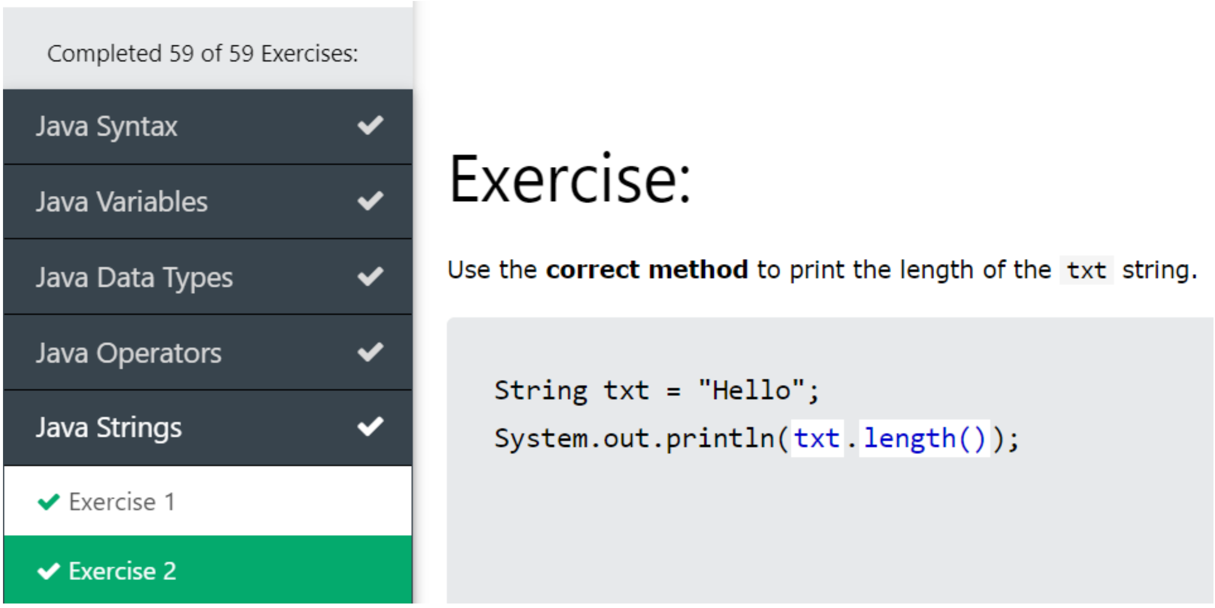
Exo sur les String :



Exercise:

Fill in the missing part to create a `greeting` variable of type `String` and assign it the value `Hello`.

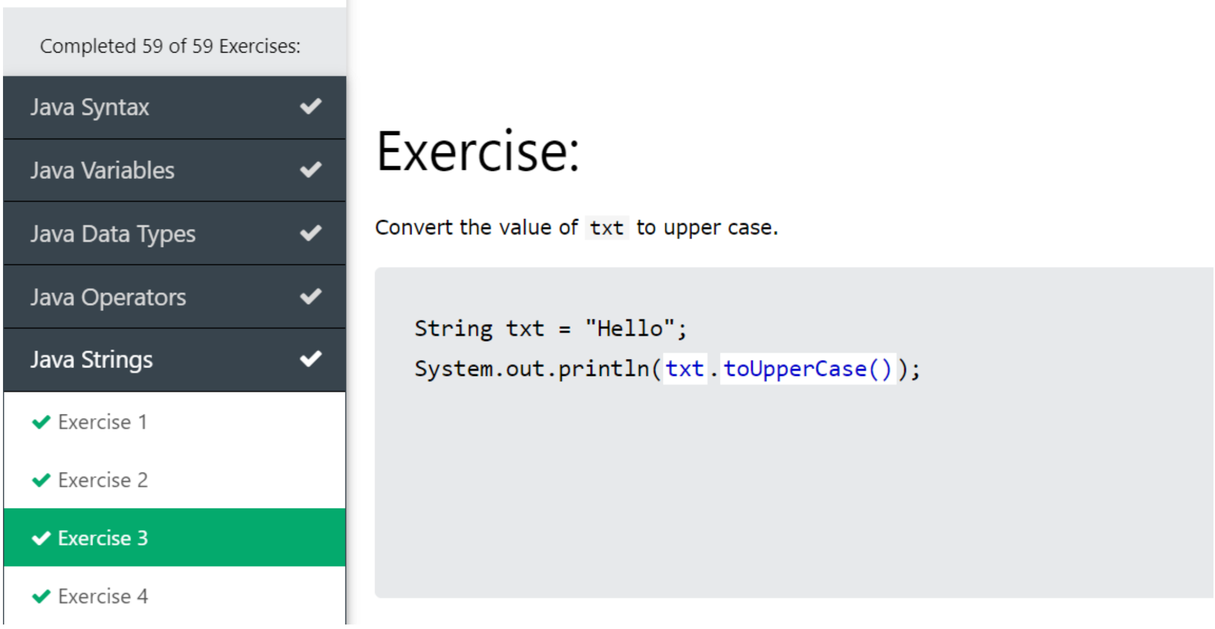
```
String greeting = "Hello";
```



Exercise:

Use the **correct method** to print the length of the `txt` string.

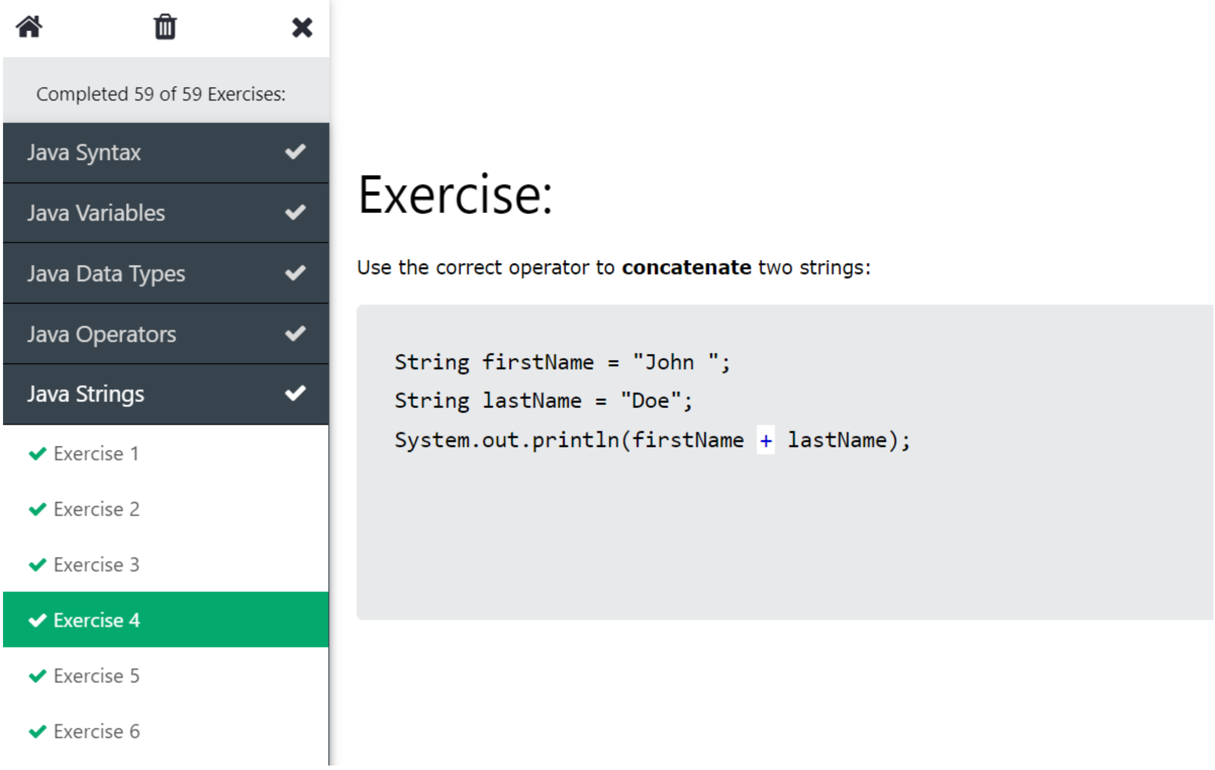
```
String txt = "Hello";
System.out.println(txt.length());
```



Exercise:

Convert the value of `txt` to upper case.

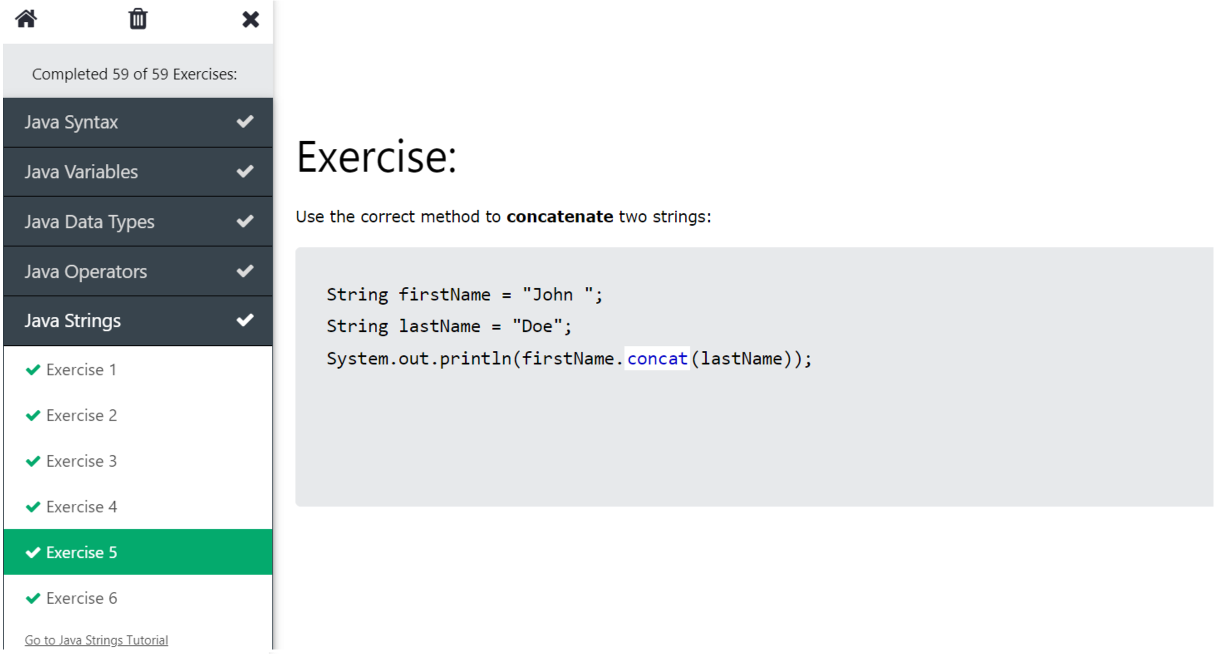
```
String txt = "Hello";
System.out.println(txt.toUpperCase());
```



Exercise:

Use the correct operator to **concatenate** two strings:

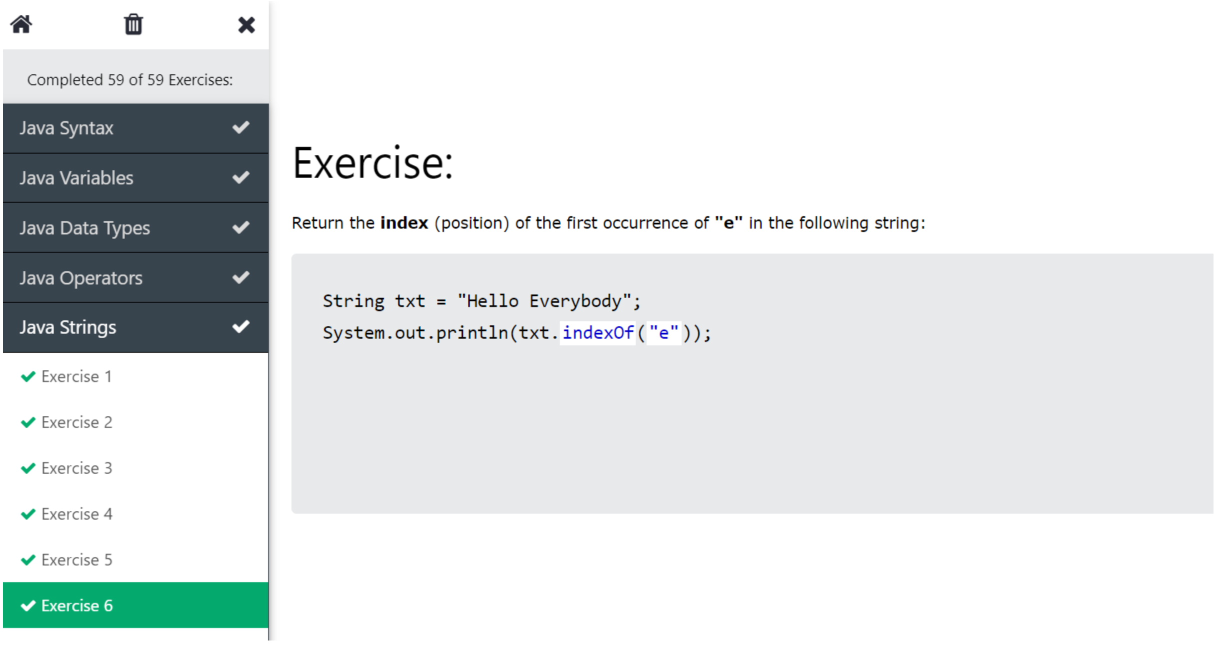
```
String firstName = "John ";
String lastName = "Doe";
System.out.println(firstName + lastName);
```



Exercise:

Use the correct method to **concatenate** two strings:

```
String firstName = "John ";
String lastName = "Doe";
System.out.println(firstName.concat(lastName));
```

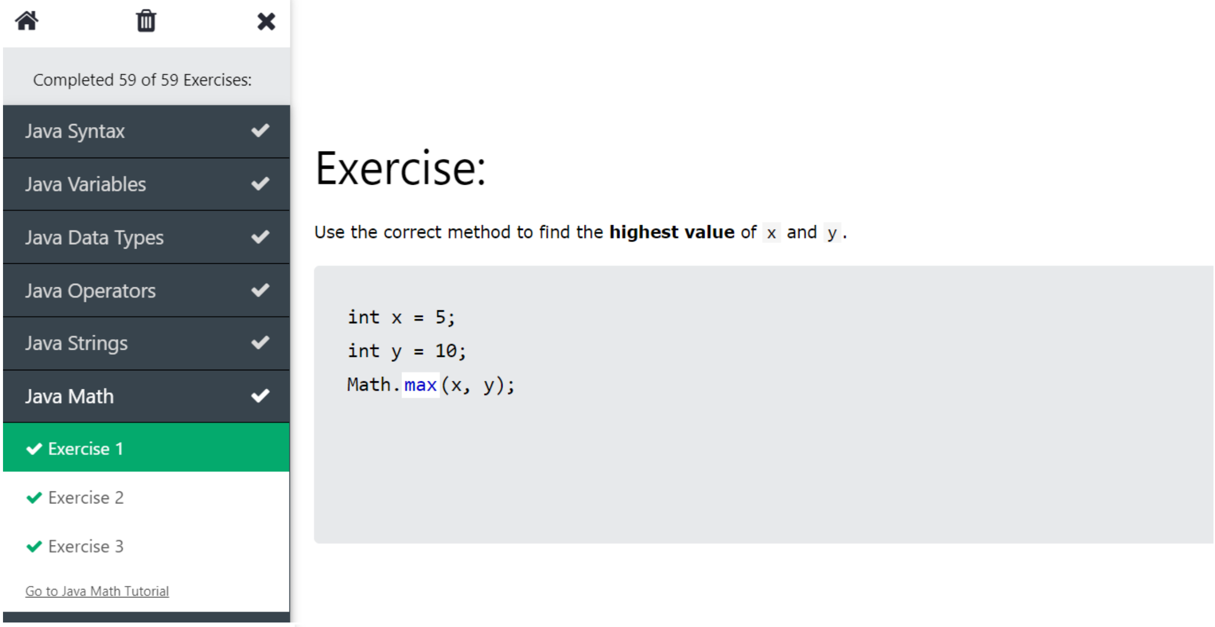


Exercise:

Return the **index** (position) of the first occurrence of **"e"** in the following string:

```
String txt = "Hello Everybody";
System.out.println(txt.indexOf("e"));
```

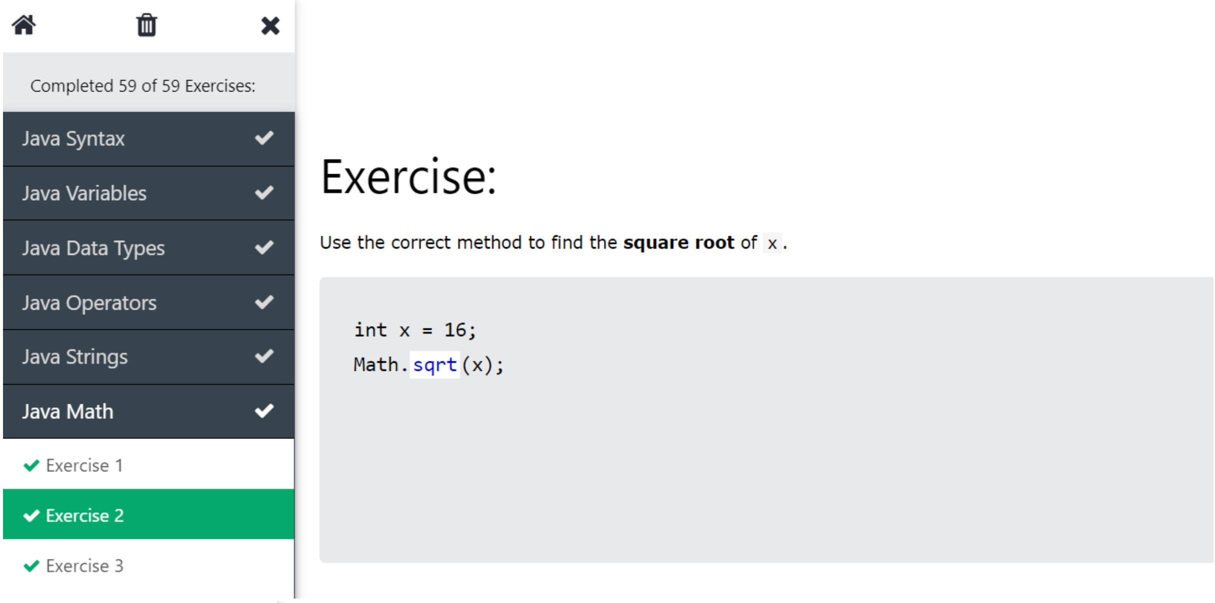
Exo sur les fonctions mathématiques :



Exercise:

Use the correct method to find the **highest value** of `x` and `y`.

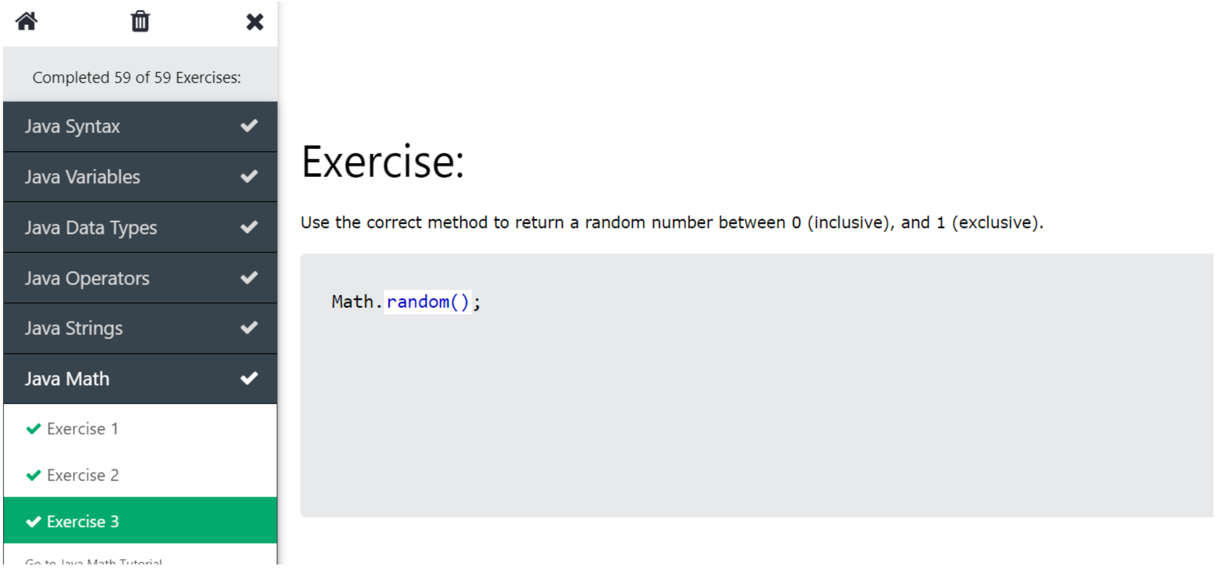
```
int x = 5;
int y = 10;
Math.max(x, y);
```



Exercise:

Use the correct method to find the **square root** of `x`.

```
int x = 16;
Math.sqrt(x);
```

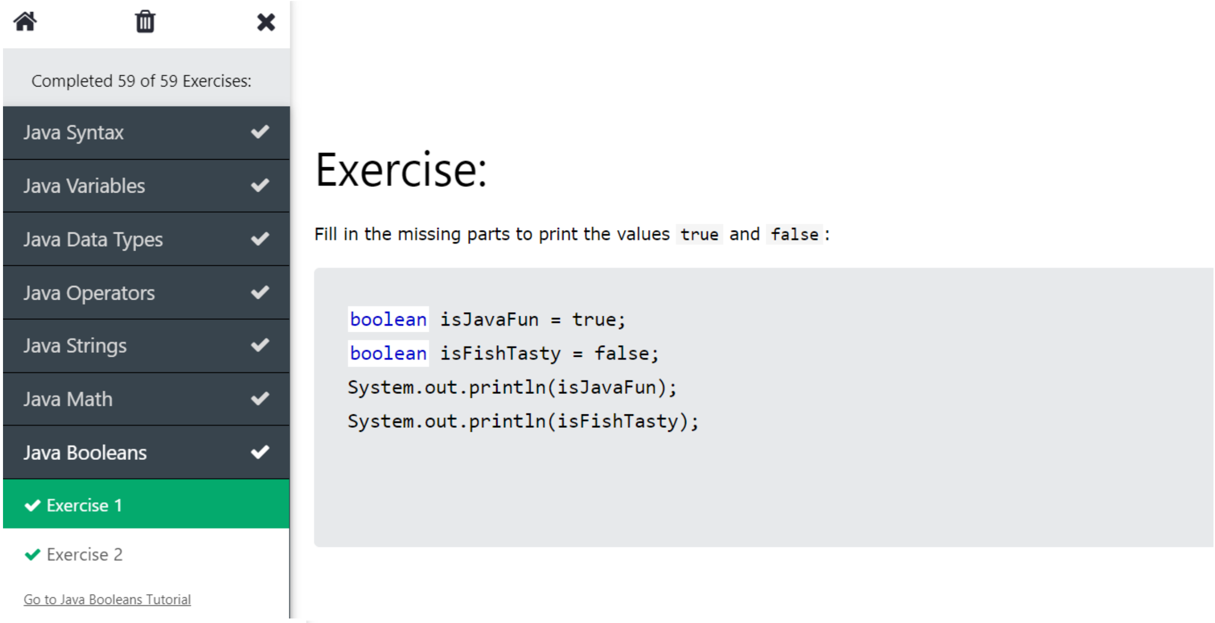


Exercise:

Use the correct method to return a random number between 0 (inclusive), and 1 (exclusive).

```
Math.random();
```

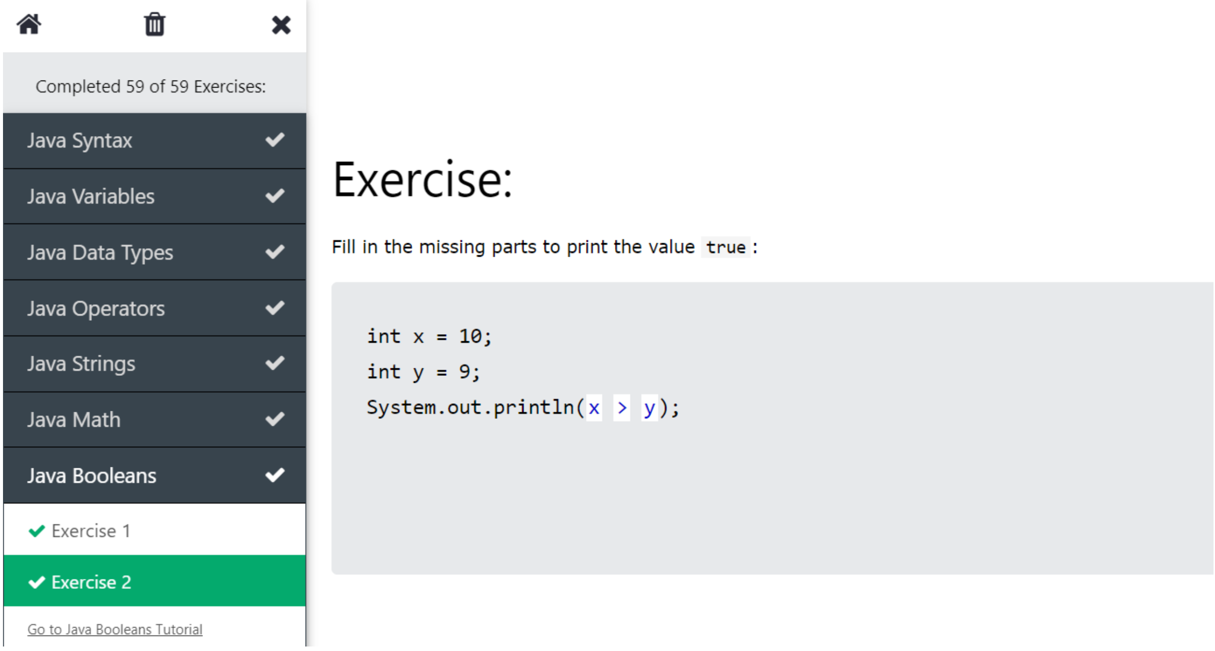
Exo sur les booléens :



Exercise:

Fill in the missing parts to print the values `true` and `false` :

```
boolean isJavaFun = true;
boolean isFishTasty = false;
System.out.println(isJavaFun);
System.out.println(isFishTasty);
```

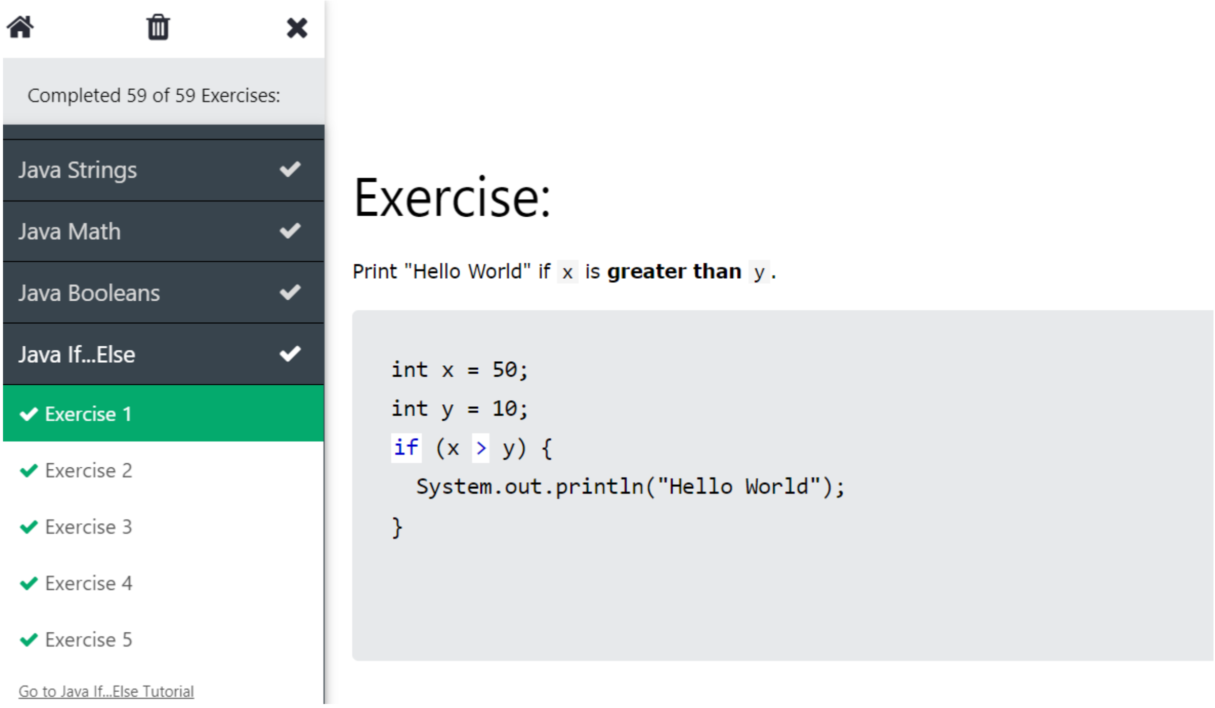


Exercise:

Fill in the missing parts to print the value `true` :

```
int x = 10;
int y = 9;
System.out.println(x > y);
```

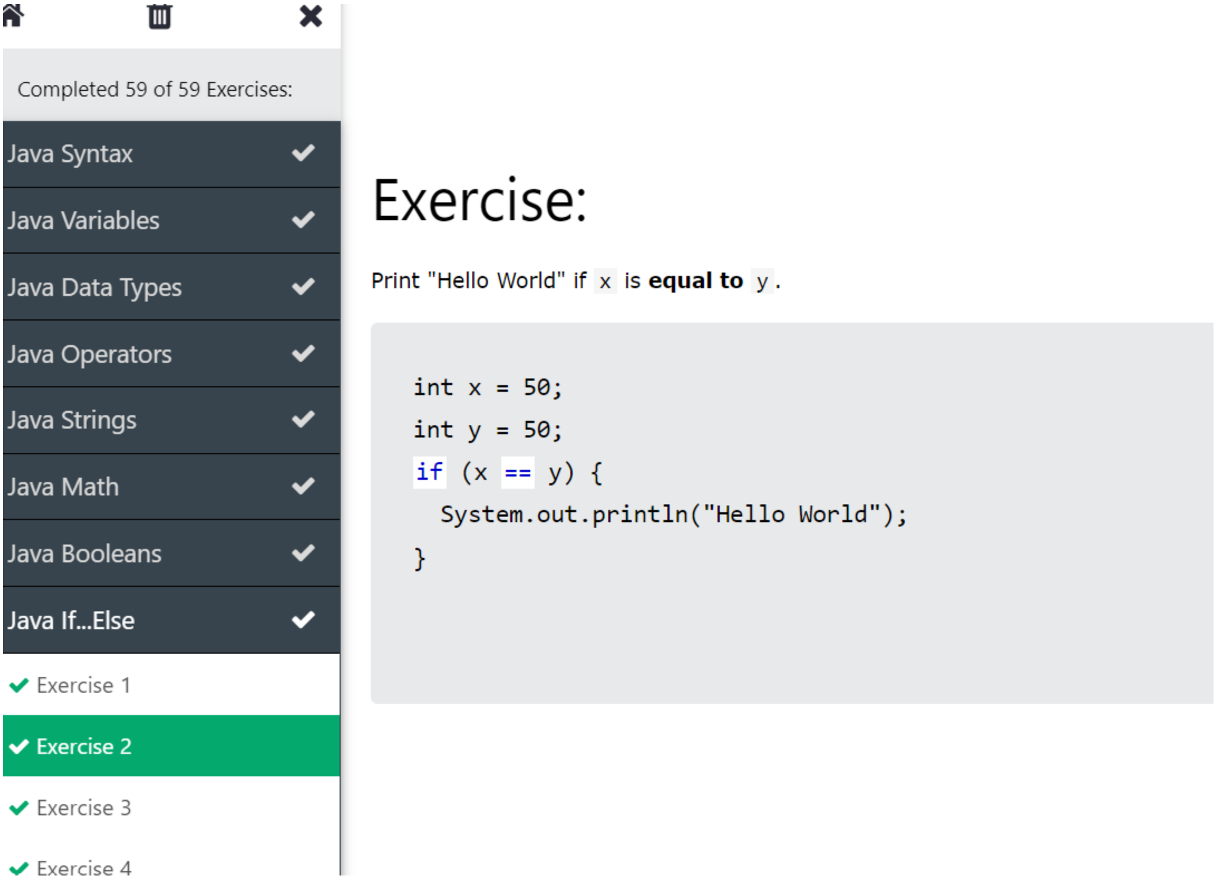
Exo sur les condtions :



Exercise:

Print "Hello World" if `x` is **greater than** `y`.

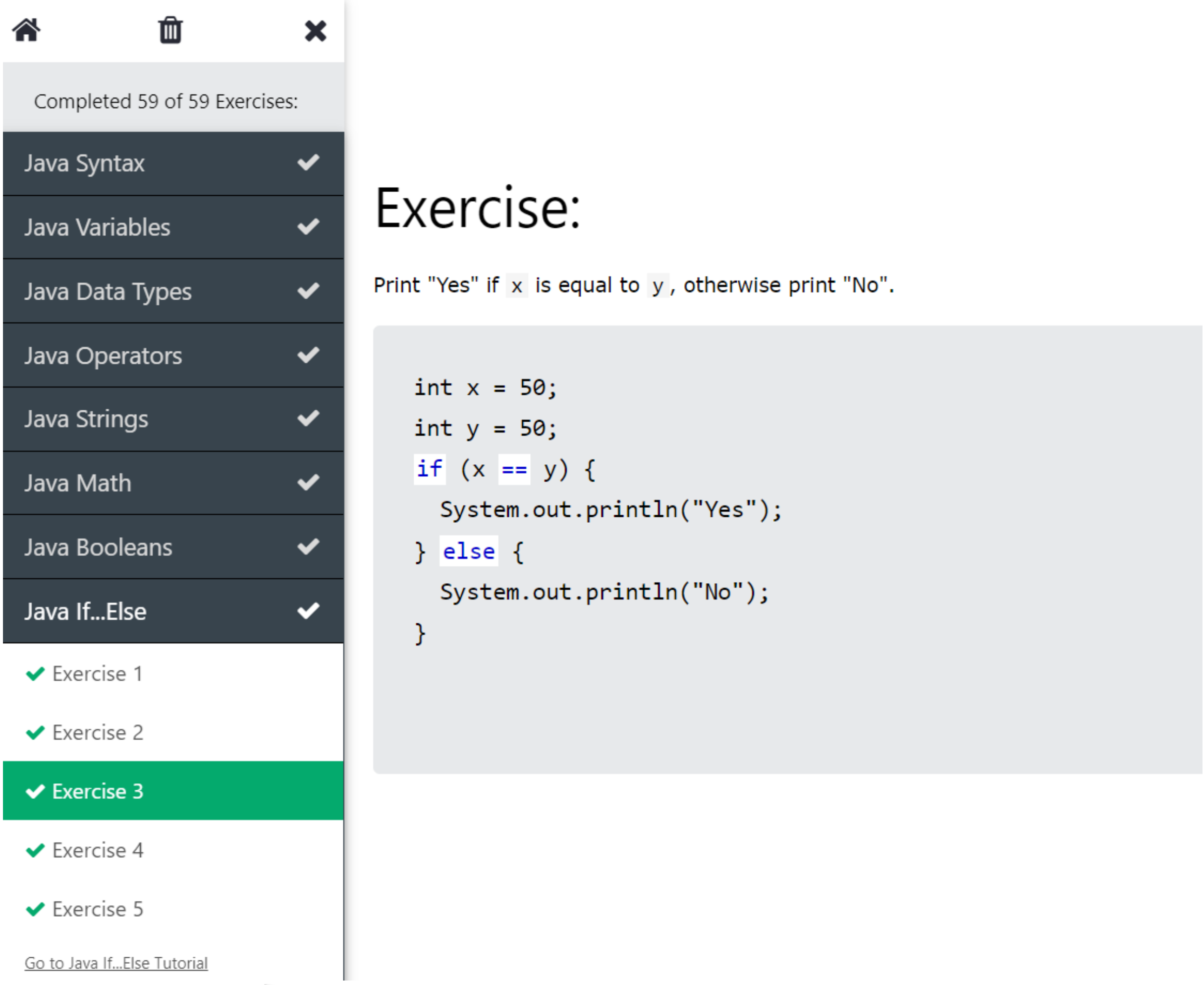
```
int x = 50;
int y = 10;
if (x > y) {
  System.out.println("Hello World");
}
```



Exercise:

Print "Hello World" if `x` is **equal to** `y`.

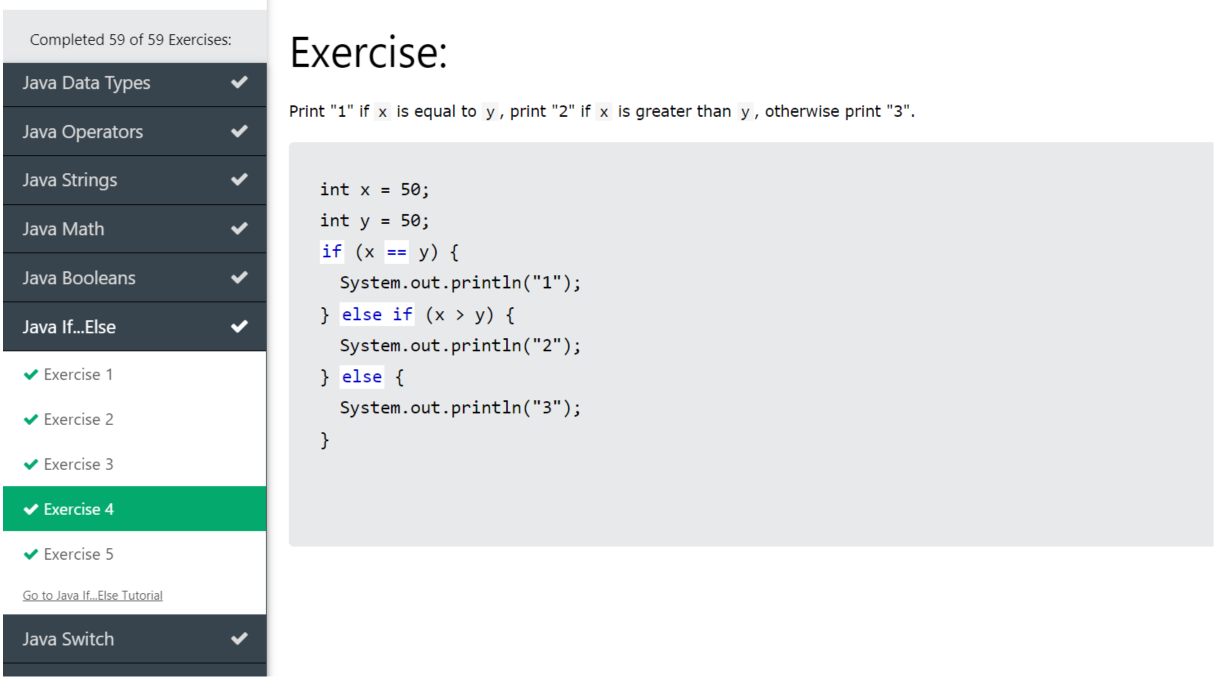
```
int x = 50;
int y = 50;
if (x == y) {
  System.out.println("Hello World");
}
```



Exercise:

Print "Yes" if `x` is equal to `y`, otherwise print "No".

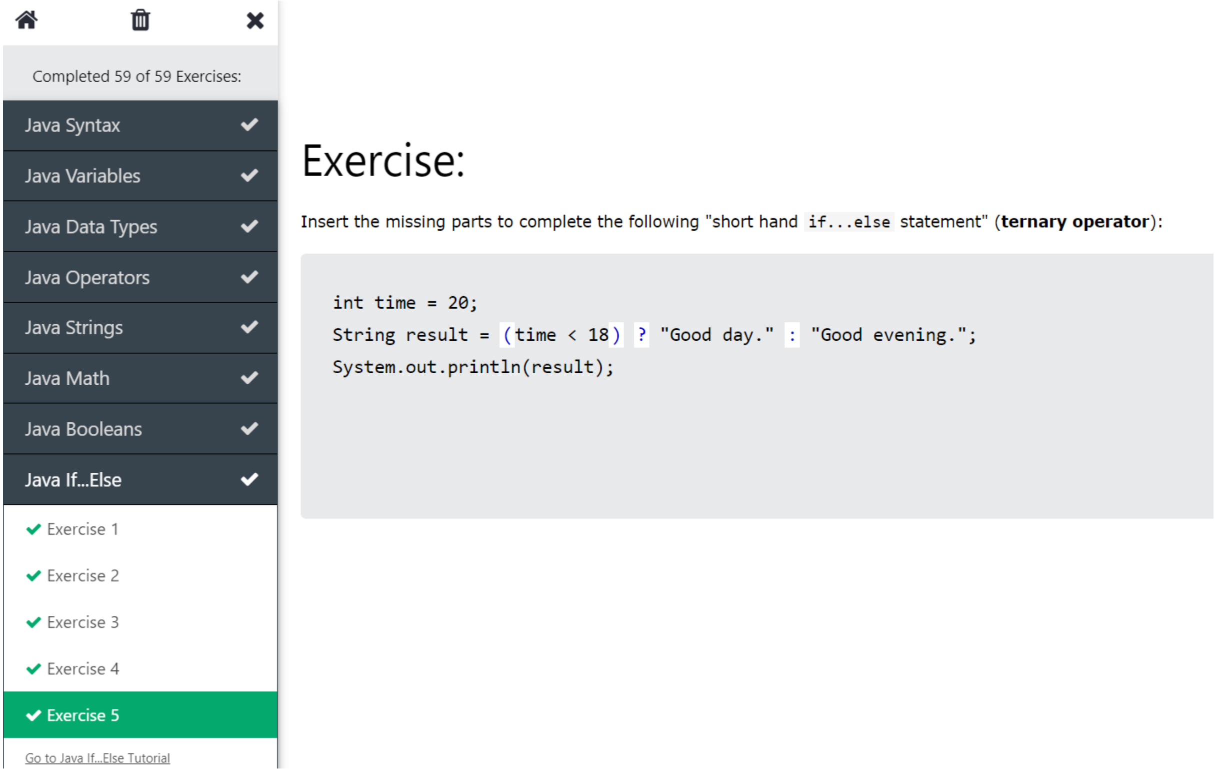
```
int x = 50;
int y = 50;
if (x == y) {
  System.out.println("Yes");
} else {
  System.out.println("No");
}
```



Exercise:

Print "1" if `x` is equal to `y`, print "2" if `x` is greater than `y`, otherwise print "3".

```
int x = 50;
int y = 50;
if (x == y) {
  System.out.println("1");
} else if (x > y) {
  System.out.println("2");
} else {
  System.out.println("3");
}
```

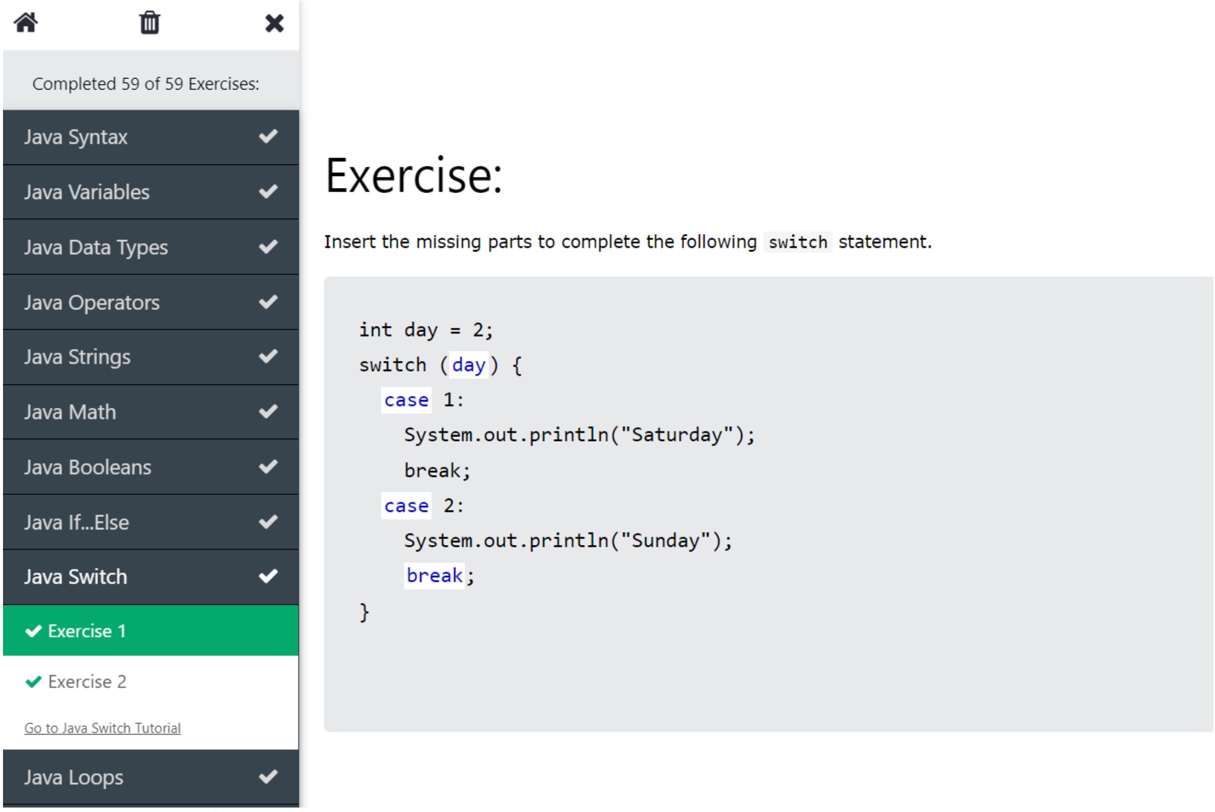


Exercise:

Insert the missing parts to complete the following "short hand `if...else` statement" (**ternary operator**):

```
int time = 20;
String result = (time < 18) ? "Good day." : "Good evening.";
System.out.println(result);
```

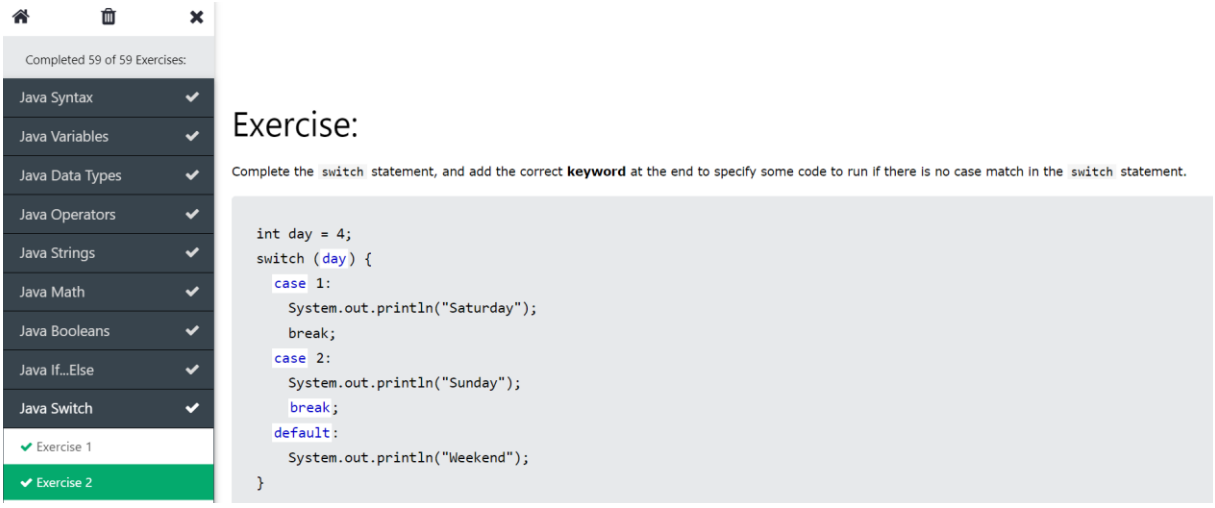
Exo sur les switch :



Exercise:

Insert the missing parts to complete the following `switch` statement.

```
int day = 2;
switch (day) {
  case 1:
    System.out.println("Saturday");
    break;
  case 2:
    System.out.println("Sunday");
    break;
}
```

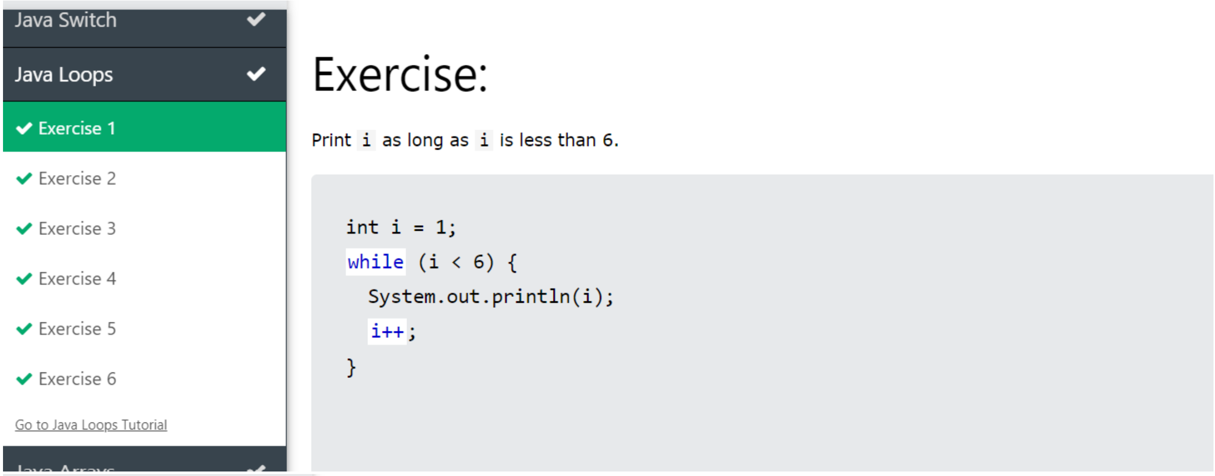


Exercise:

Complete the `switch` statement, and add the correct **keyword** at the end to specify some code to run if there is no case match in the `switch` statement.

```
int day = 4;
switch (day) {
  case 1:
    System.out.println("Saturday");
    break;
  case 2:
    System.out.println("Sunday");
    break;
  default:
    System.out.println("Weekend");
}
```

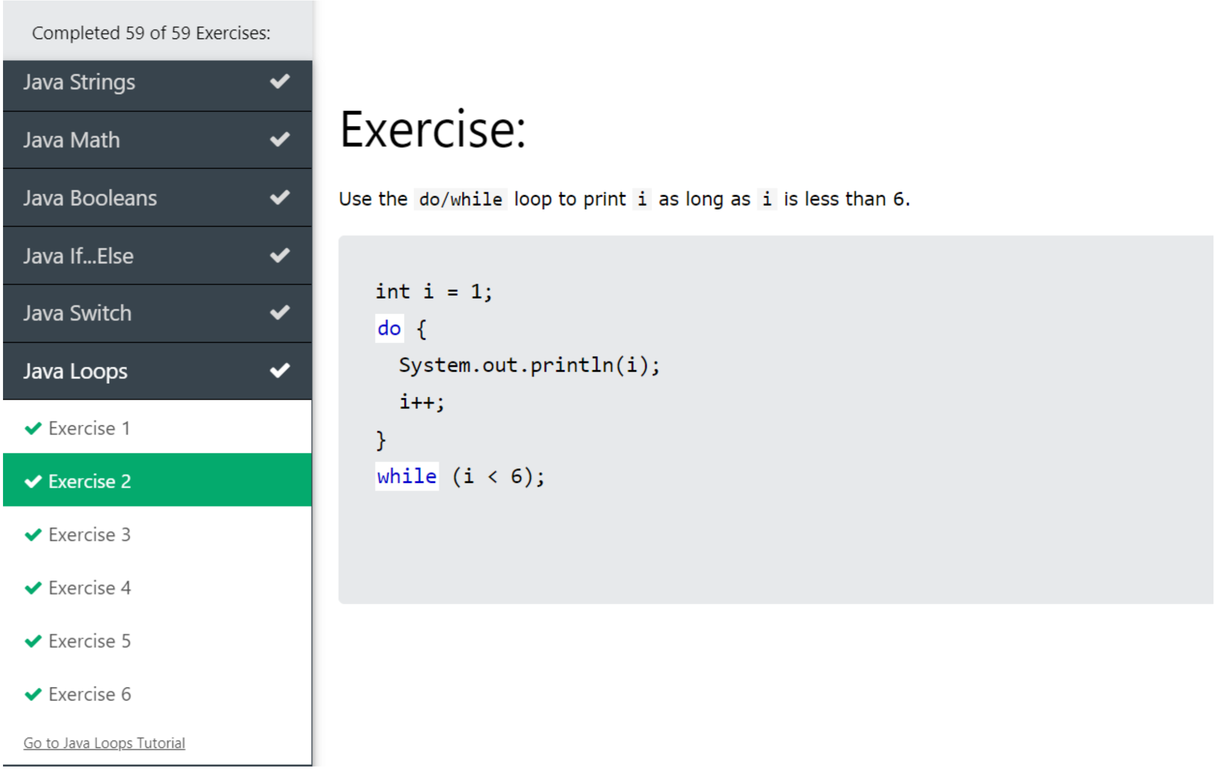
Exo sur les boucles :



Exercise:

Print i as long as i is less than 6.

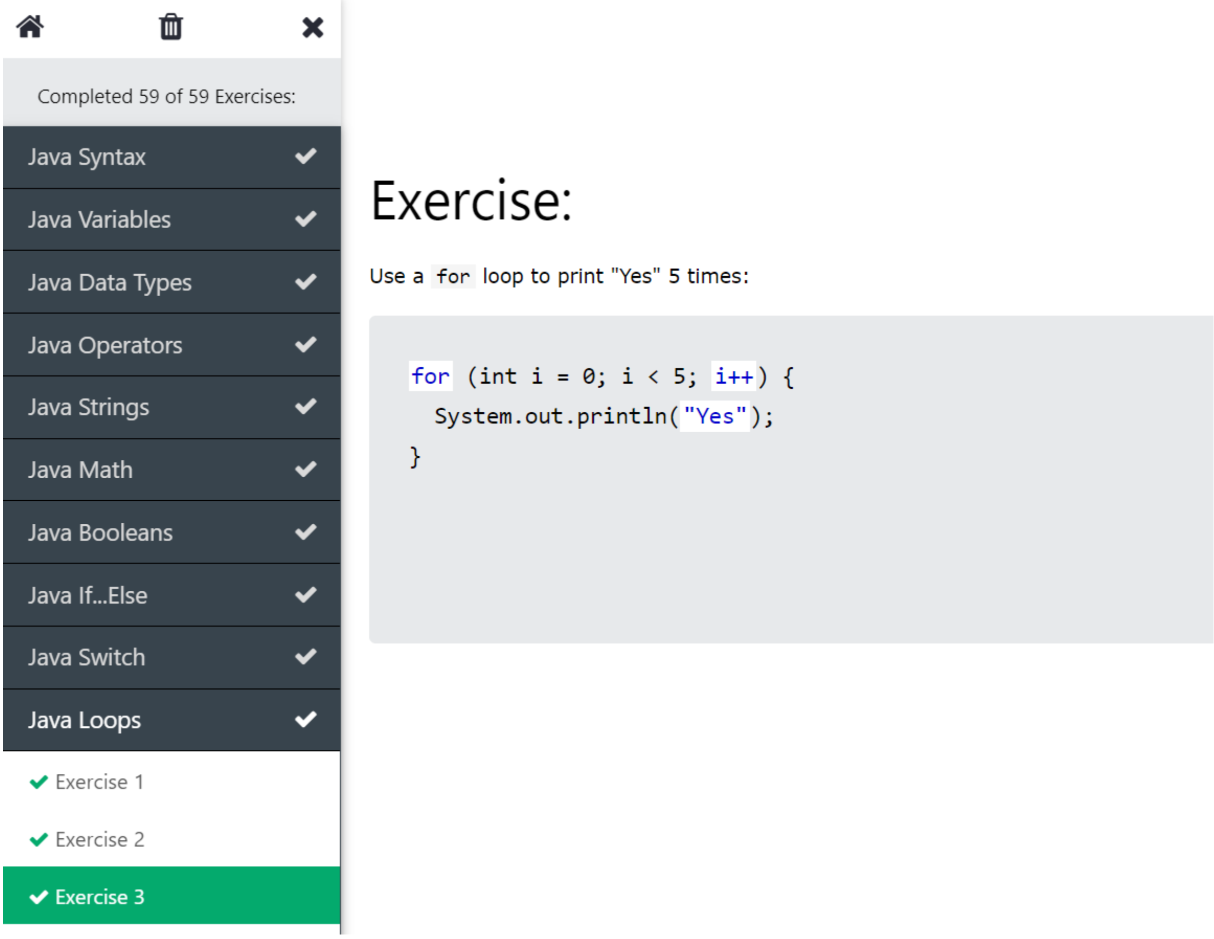
```
int i = 1;
while (i < 6) {
  System.out.println(i);
  i++;
}
```



Exercise:

Use the do/while loop to print i as long as i is less than 6.

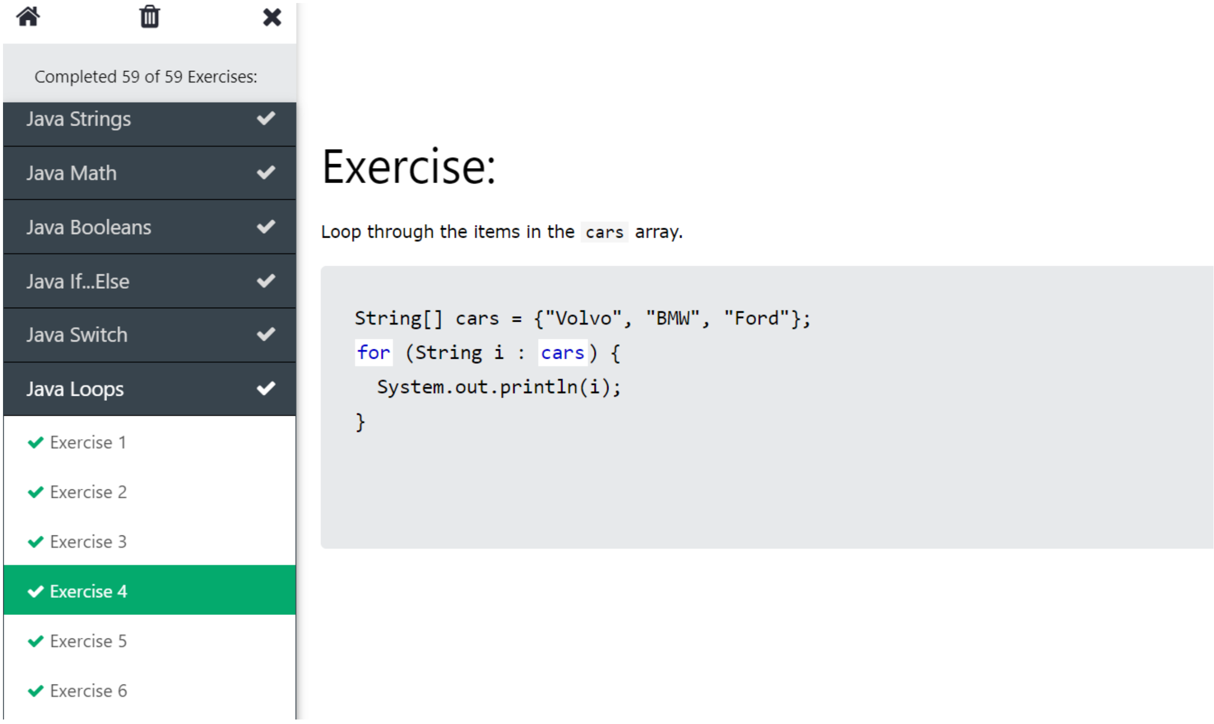
```
int i = 1;
do {
  System.out.println(i);
  i++;
}
while (i < 6);
```



Exercise:

Use a `for` loop to print "Yes" 5 times:

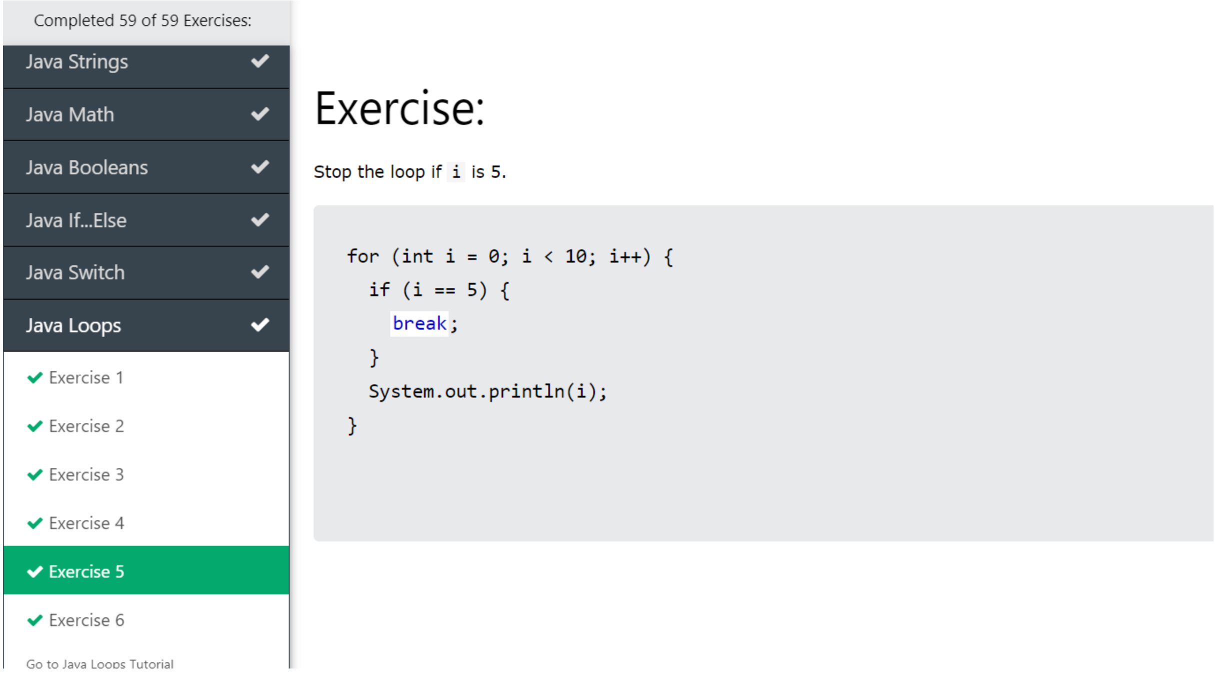
```
for (int i = 0; i < 5; i++) {
  System.out.println("Yes");
}
```



Exercise:

Loop through the items in the `cars` array.

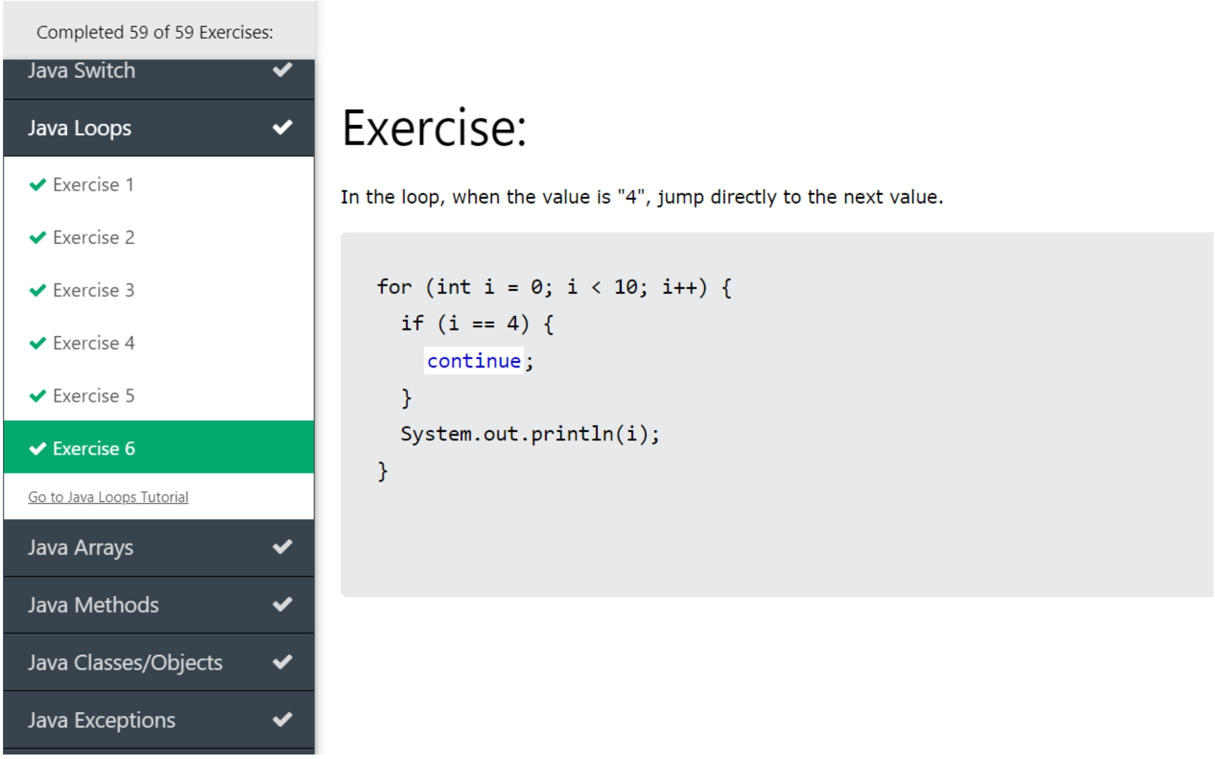
```
String[] cars = {"Volvo", "BMW", "Ford"};
for (String i : cars) {
  System.out.println(i);
}
```



Exercise:

Stop the loop if `i` is 5.

```
for (int i = 0; i < 10; i++) {
  if (i == 5) {
    break;
  }
  System.out.println(i);
}
```

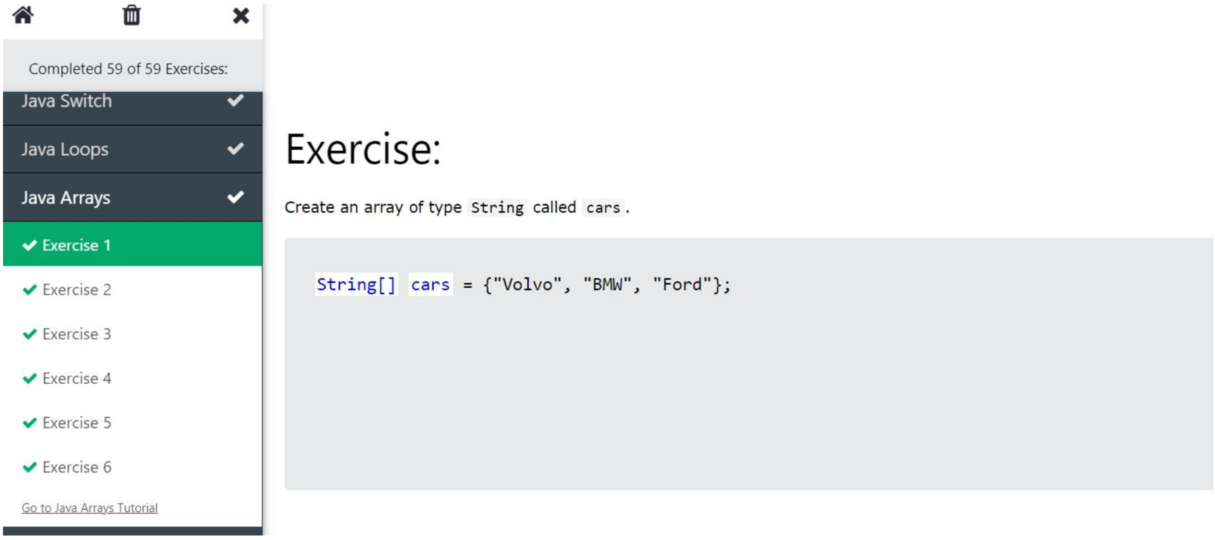


Exercise:

In the loop, when the value is "4", jump directly to the next value.

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  System.out.println(i);
}
```

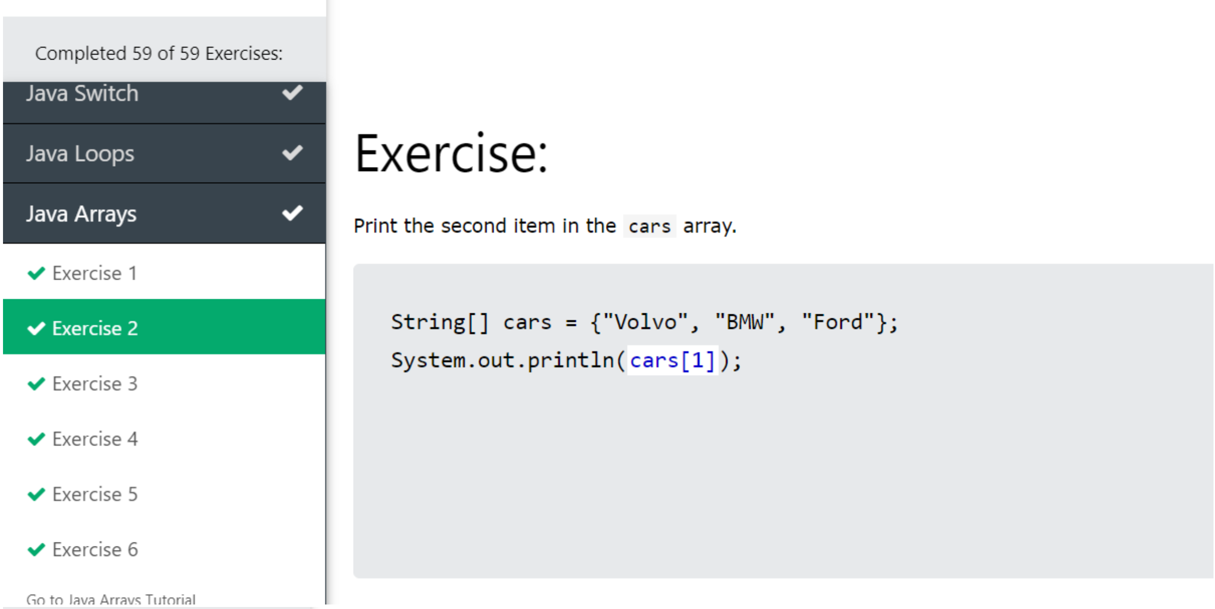
Exo sur les tableaux :



Exercise:

Create an array of type `String` called `cars`.

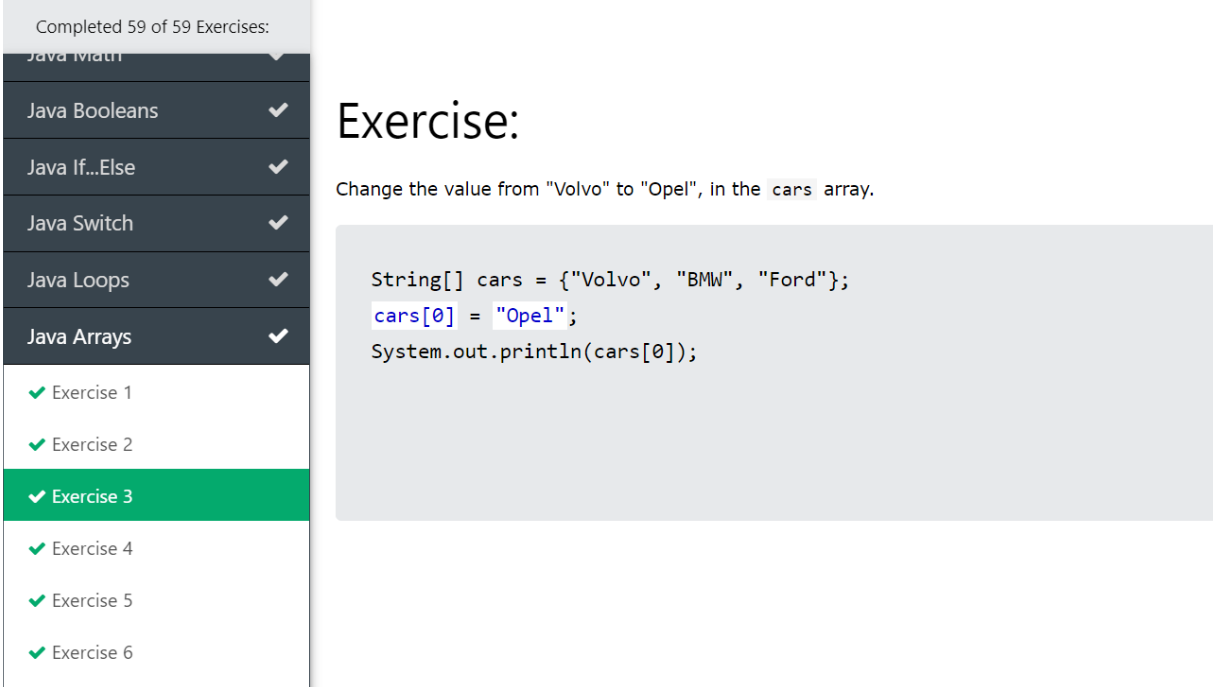
```
String[] cars = {"Volvo", "BMW", "Ford"};
```



Exercise:

Print the second item in the `cars` array.

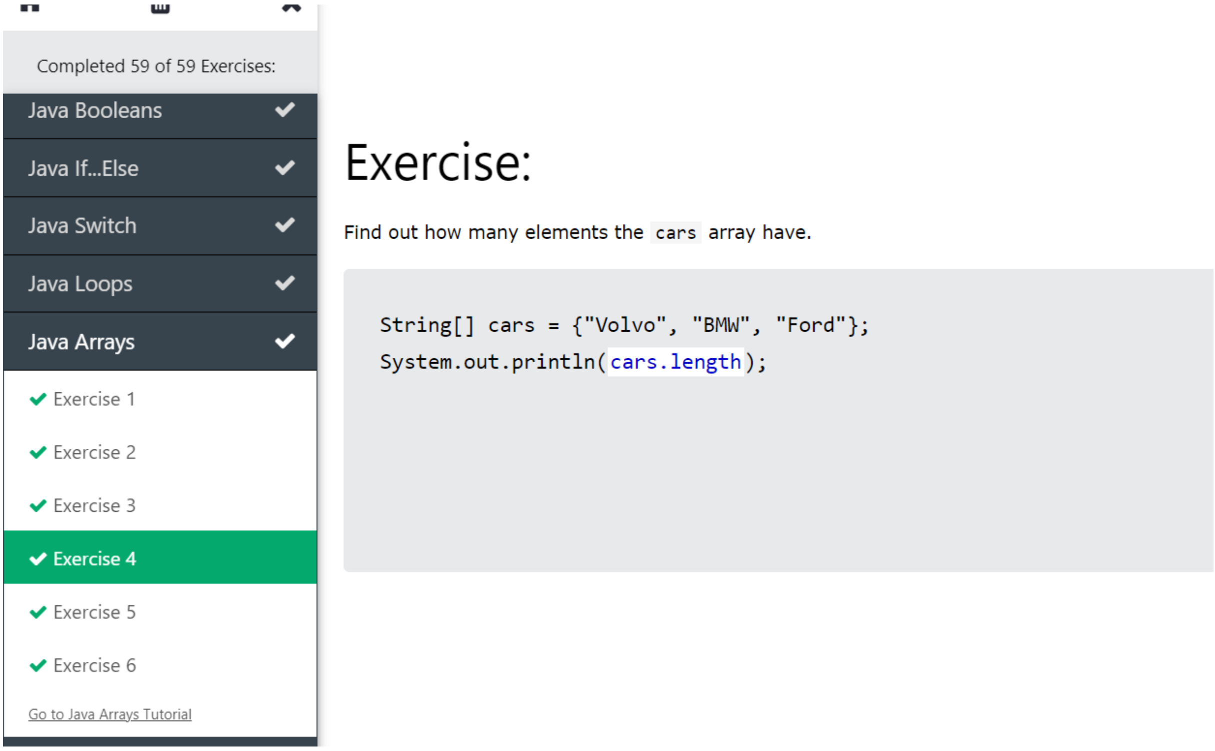
```
String[] cars = {"Volvo", "BMW", "Ford"};
System.out.println(cars[1]);
```



Exercise:

Change the value from "Volvo" to "Opel", in the `cars` array.

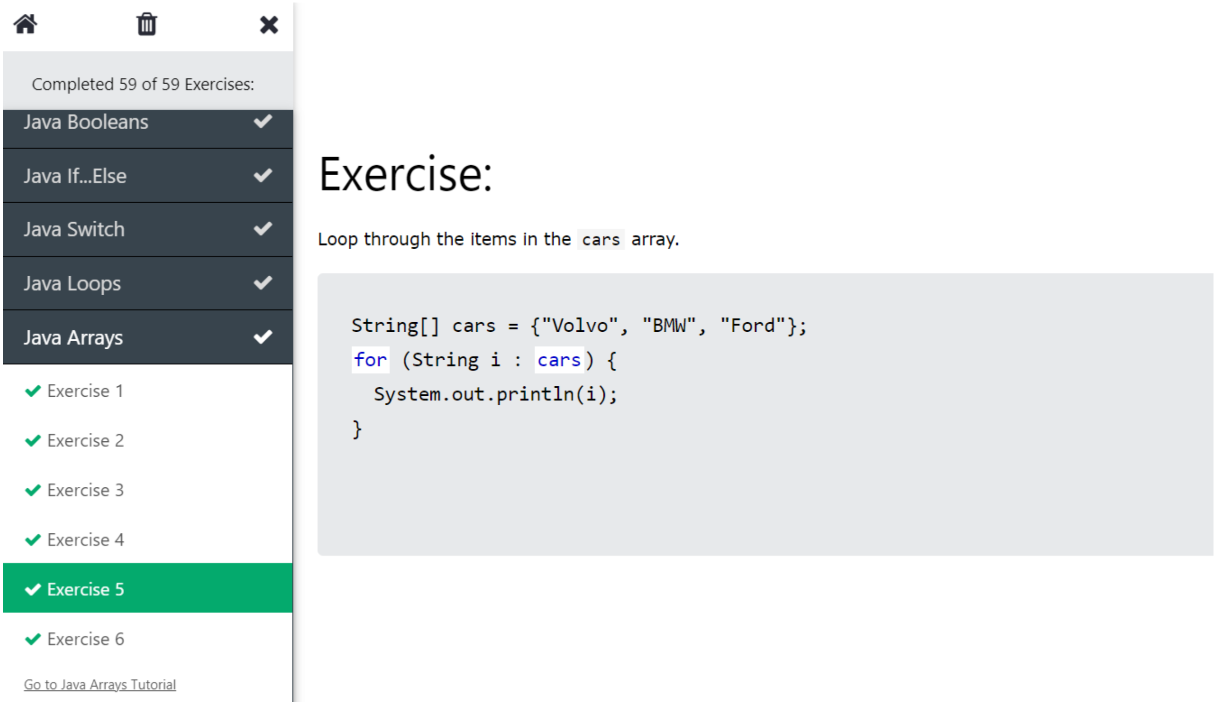
```
String[] cars = {"Volvo", "BMW", "Ford"};
cars[0] = "Opel";
System.out.println(cars[0]);
```



Exercise:

Find out how many elements the `cars` array have.

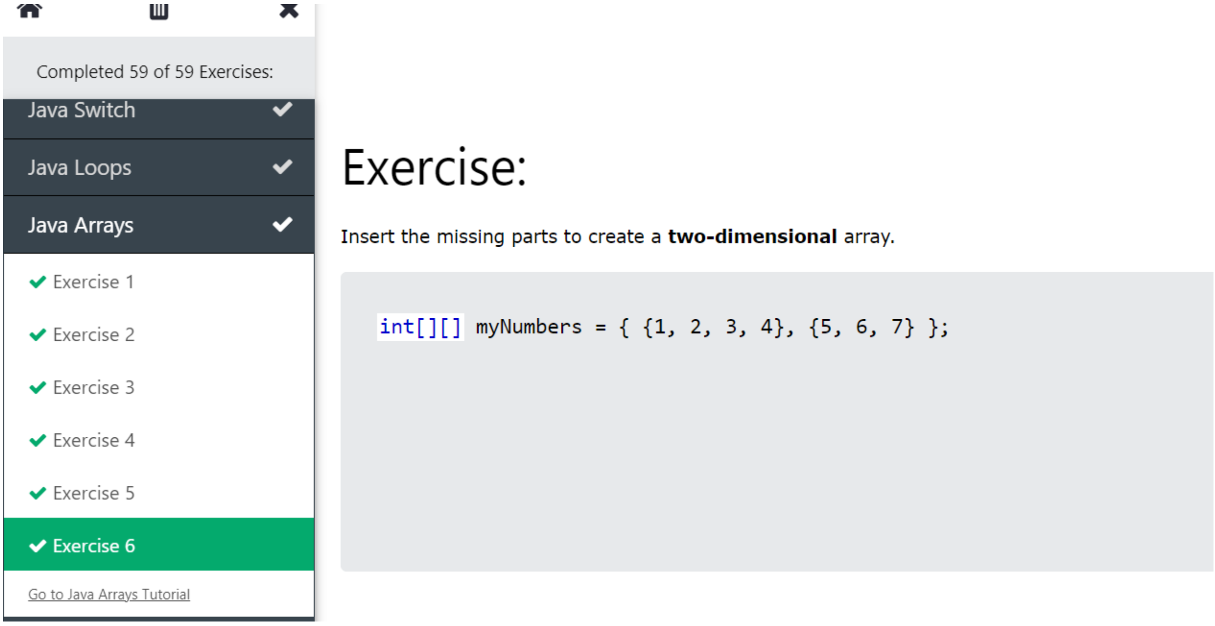
```
String[] cars = {"Volvo", "BMW", "Ford"};
System.out.println(cars.length);
```



Exercise:

Loop through the items in the `cars` array.

```
String[] cars = {"Volvo", "BMW", "Ford"};
for (String i : cars) {
  System.out.println(i);
}
```

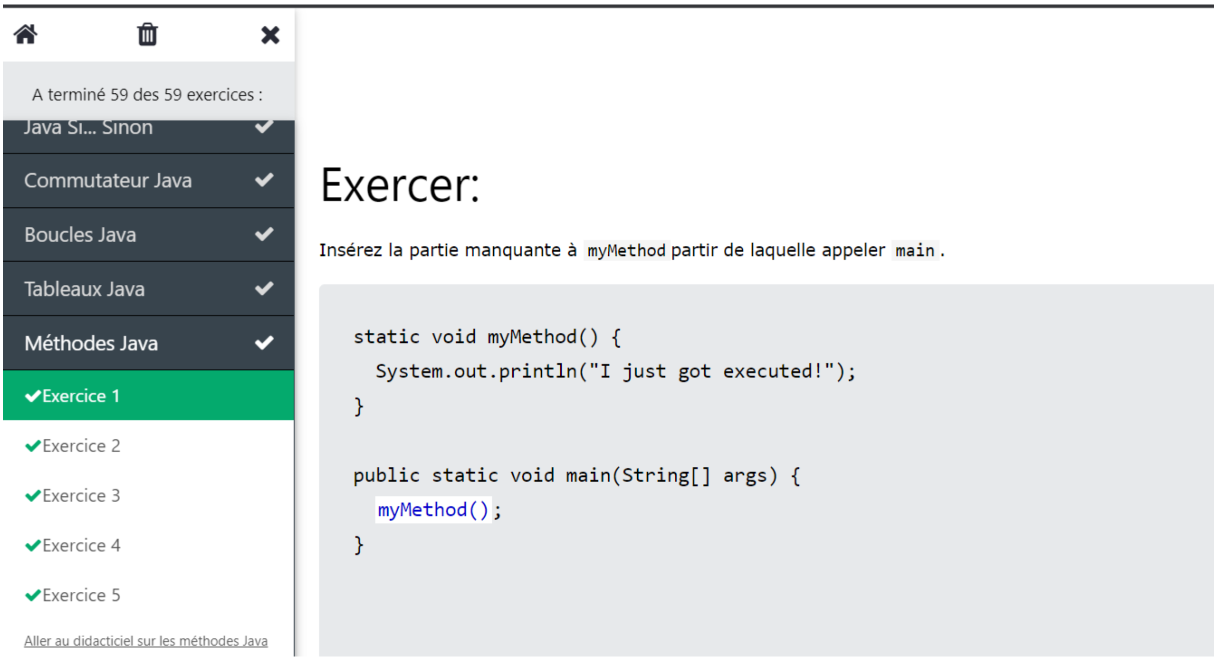


Exercise:

Insert the missing parts to create a **two-dimensional** array.

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
```

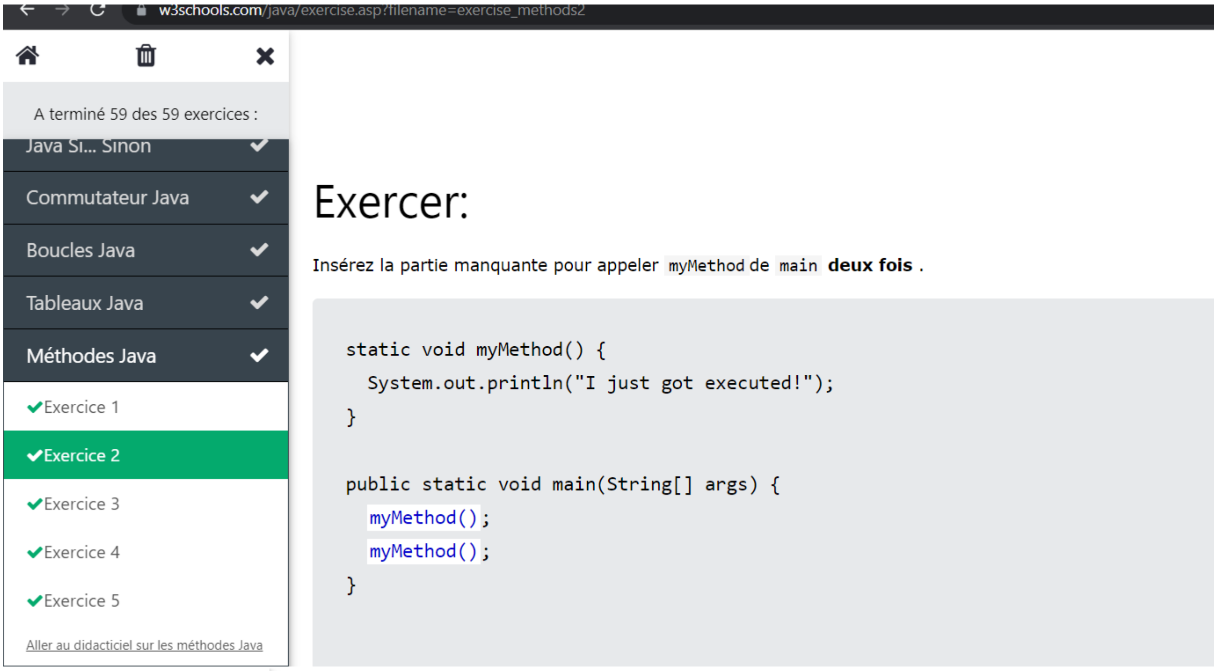
Exo sur les méthodes :



Exercer:

Insérez la partie manquante à `myMethod` partir de laquelle appeler `main` .

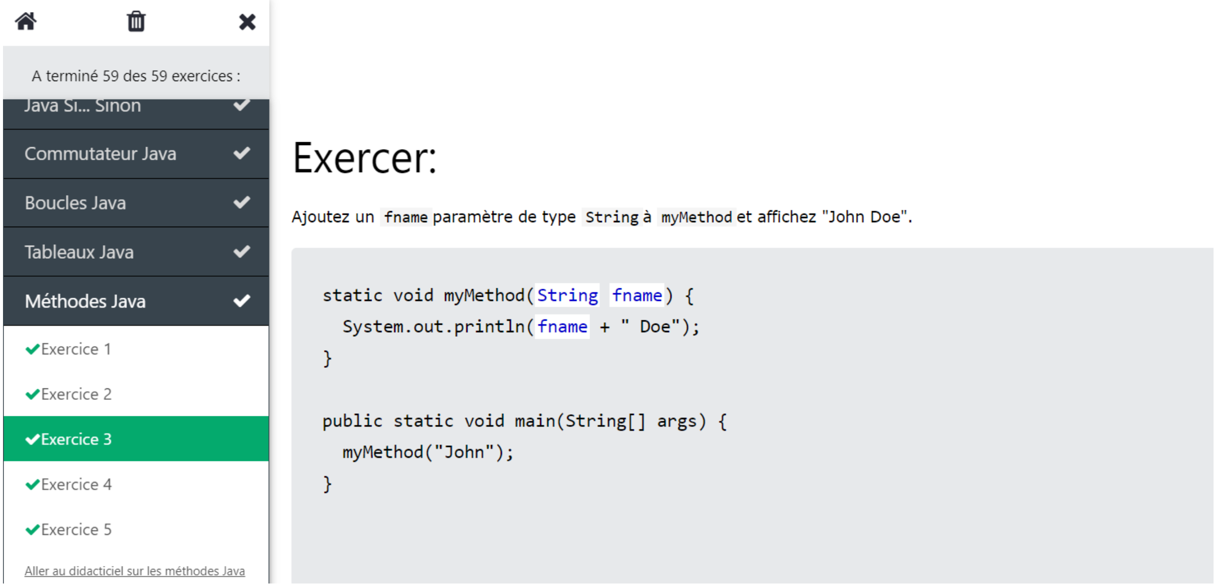
```
static void myMethod() {
  System.out.println("I just got executed!");
}

public static void main(String[] args) {
  myMethod();
}
```



Exercer:

Insérez la partie manquante pour appeler `myMethod` de `main` **deux fois** .

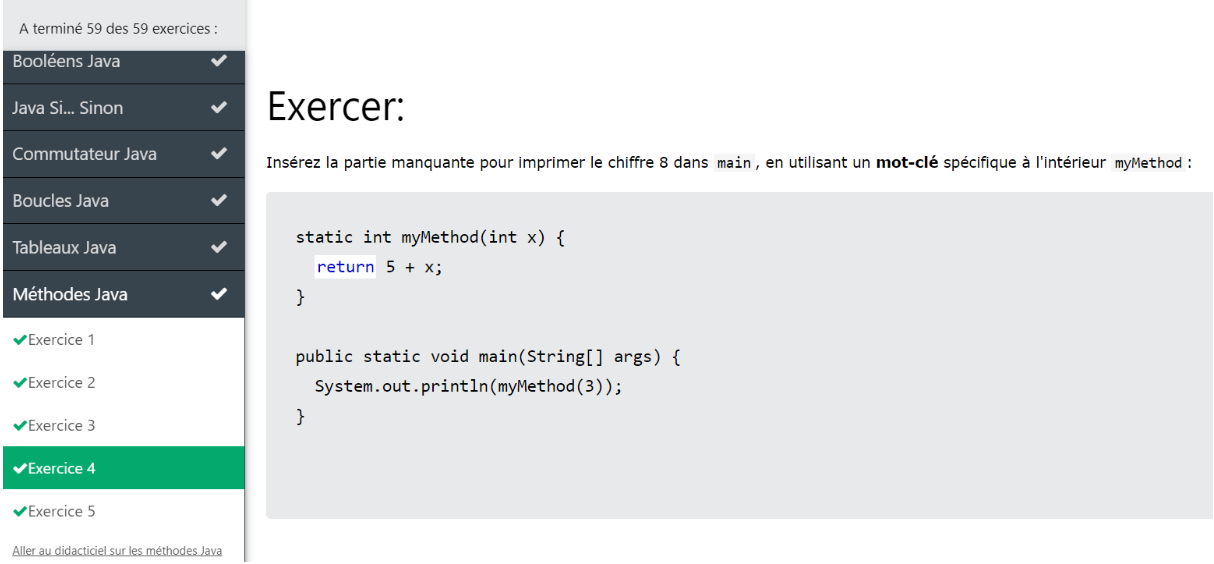
```
static void myMethod() {
  System.out.println("I just got executed!");
}

public static void main(String[] args) {
  myMethod();
  myMethod();
}
```



Exercer:

Ajoutez un `fname` paramètre de type `String` à `myMethod` et affichez "John Doe".

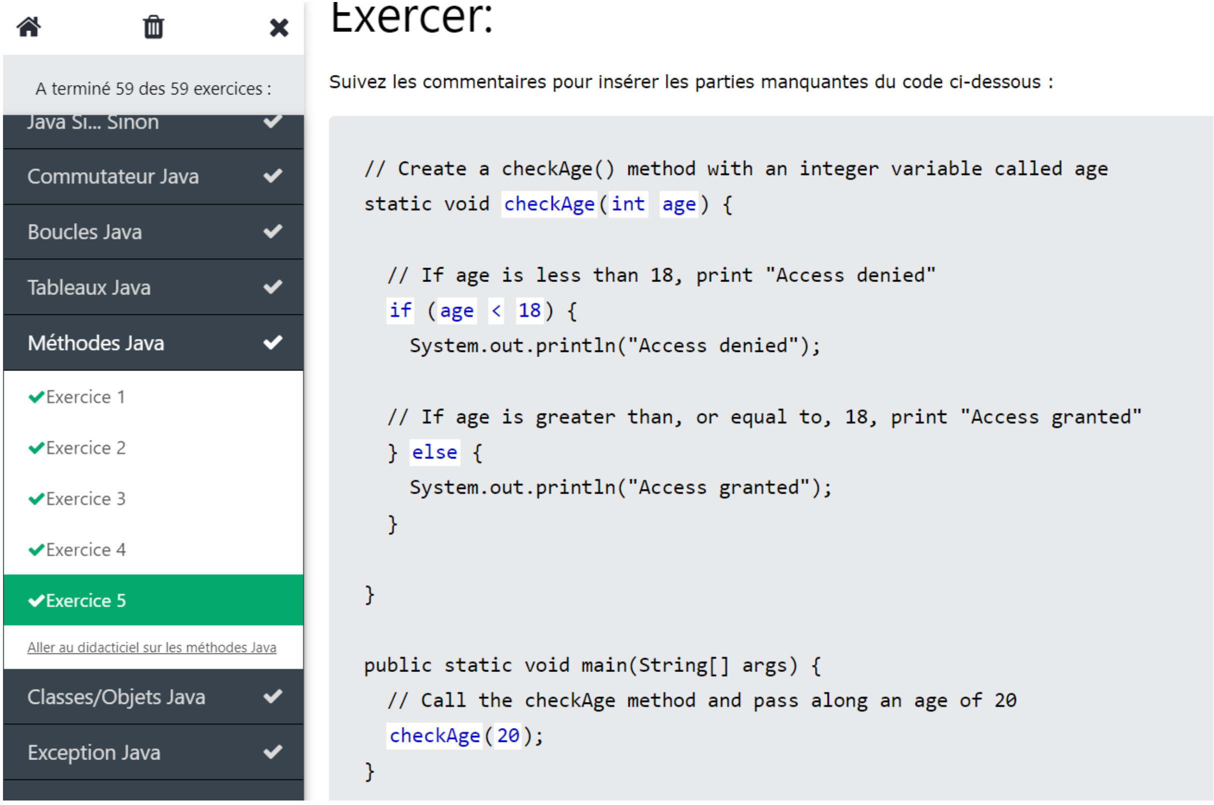
```
static void myMethod(String fname) {
  System.out.println(fname + " Doe");
}

public static void main(String[] args) {
  myMethod("John");
}
```



Exercer:

Insérez la partie manquante pour imprimer le chiffre 8 dans `main` , en utilisant un **mot-clé** spécifique à l'intérieur `myMethod` :

```
static int myMethod(int x) {
  return 5 + x;
}

public static void main(String[] args) {
  System.out.println(myMethod(3));
}
```

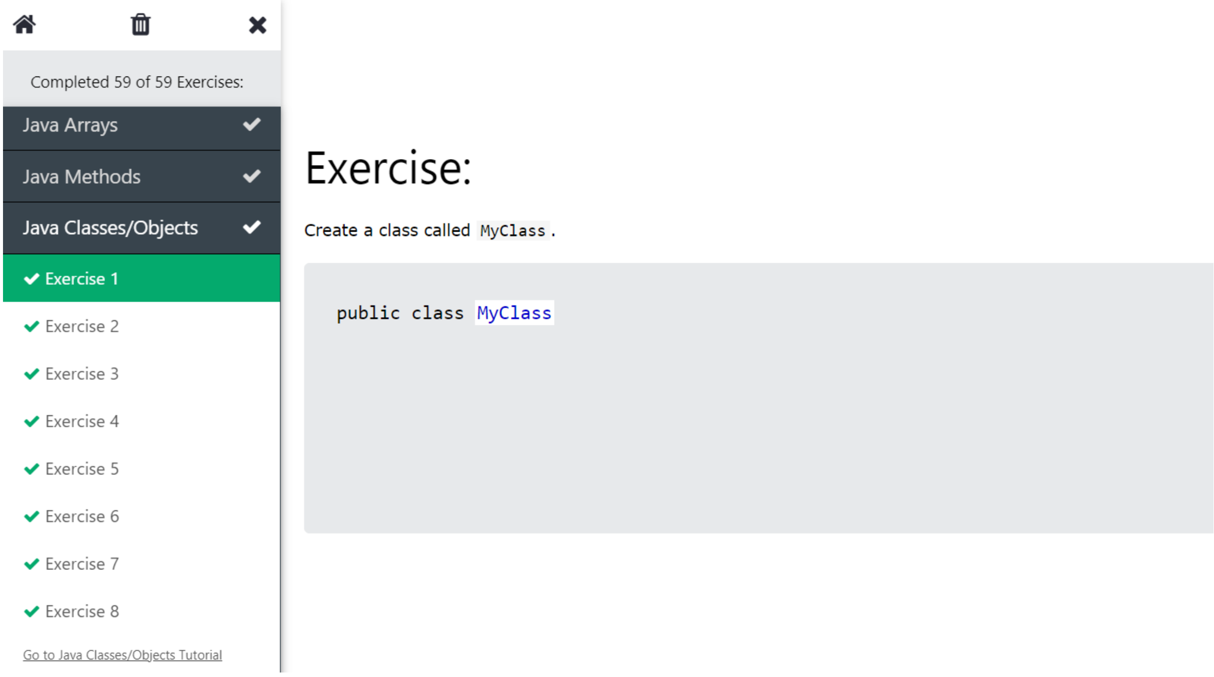


Exercer:

Suivez les commentaires pour insérer les parties manquantes du code ci-dessous :

```
// Create a checkAge() method with an integer variable called age
static void checkAge(int age) {

  // If age is less than 18, print "Access denied"
  if (age < 18) {
    System.out.println("Access denied");

  // If age is greater than, or equal to, 18, print "Access granted"
  } else {
    System.out.println("Access granted");
  }

}

public static void main(String[] args) {
  // Call the checkAge method and pass along an age of 20
  checkAge(20);
}
```

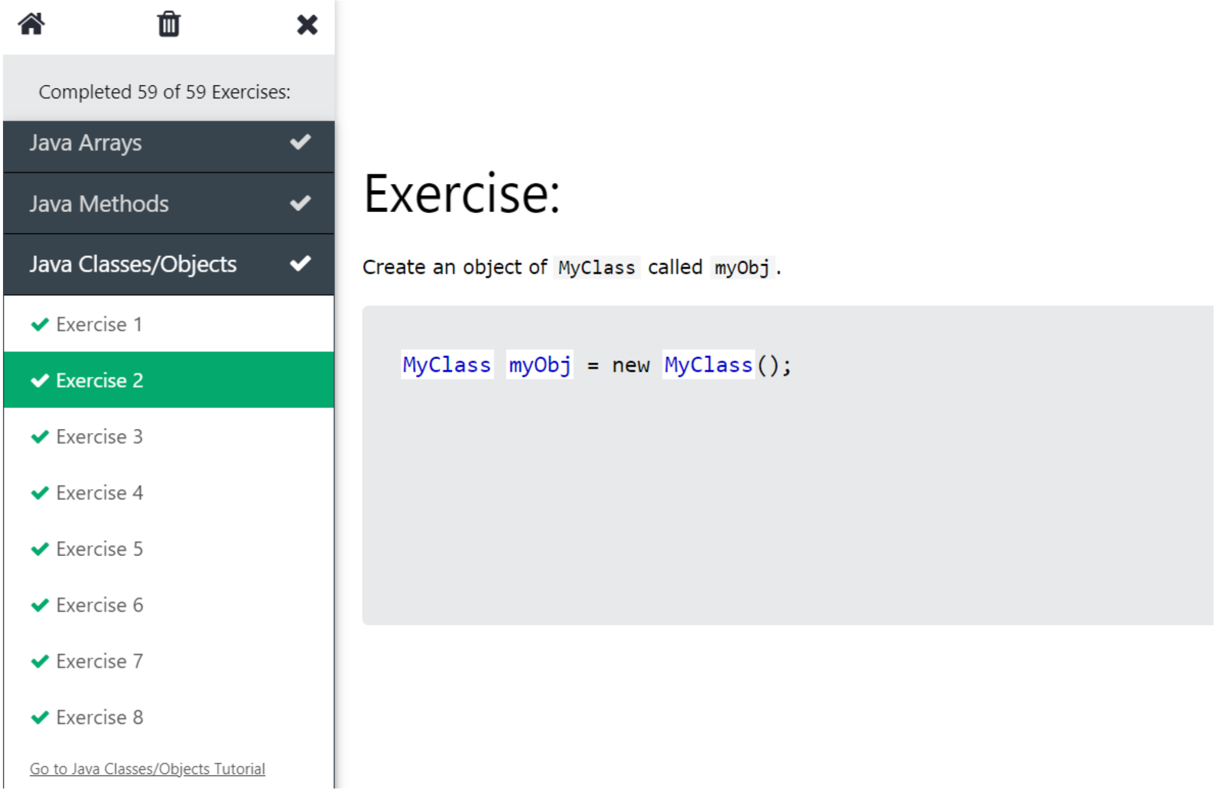
Exo sur les objets/classe :



Exercise:

Create a class called `MyClass` .

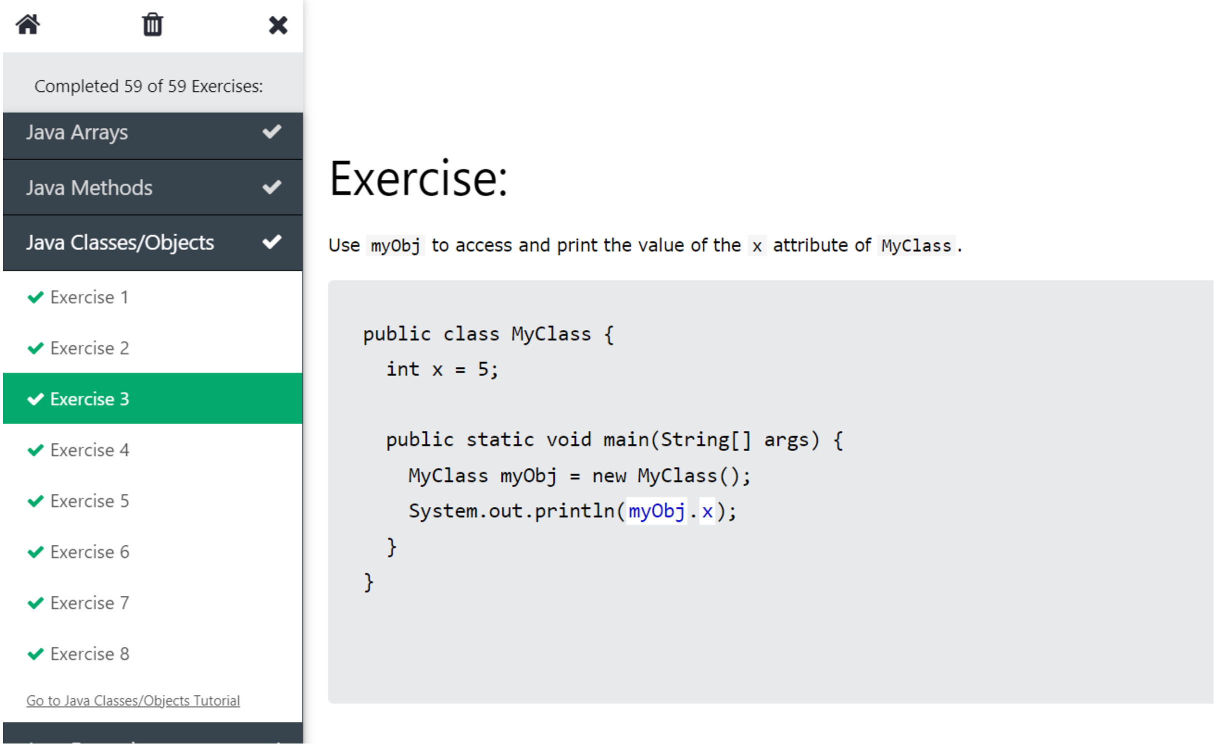
```
public class MyClass
```



Exercise:

Create an object of `MyClass` called `myObj` .

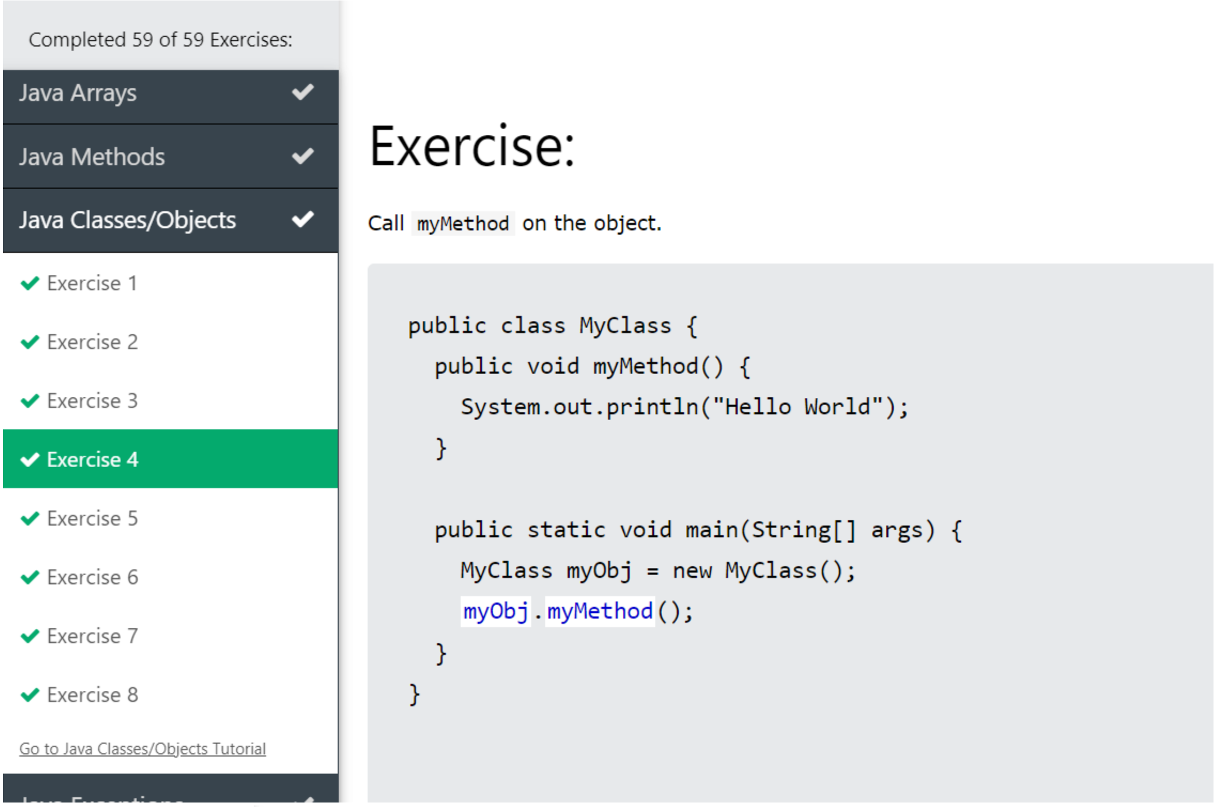
```
MyClass myObj = new MyClass();
```



Exercise:

Use `myObj` to access and print the value of the `x` attribute of `MyClass`.

```
public class MyClass {
  int x = 5;

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    System.out.println(myObj.x);
  }
}
```



Completed 59 of 59 Exercises:

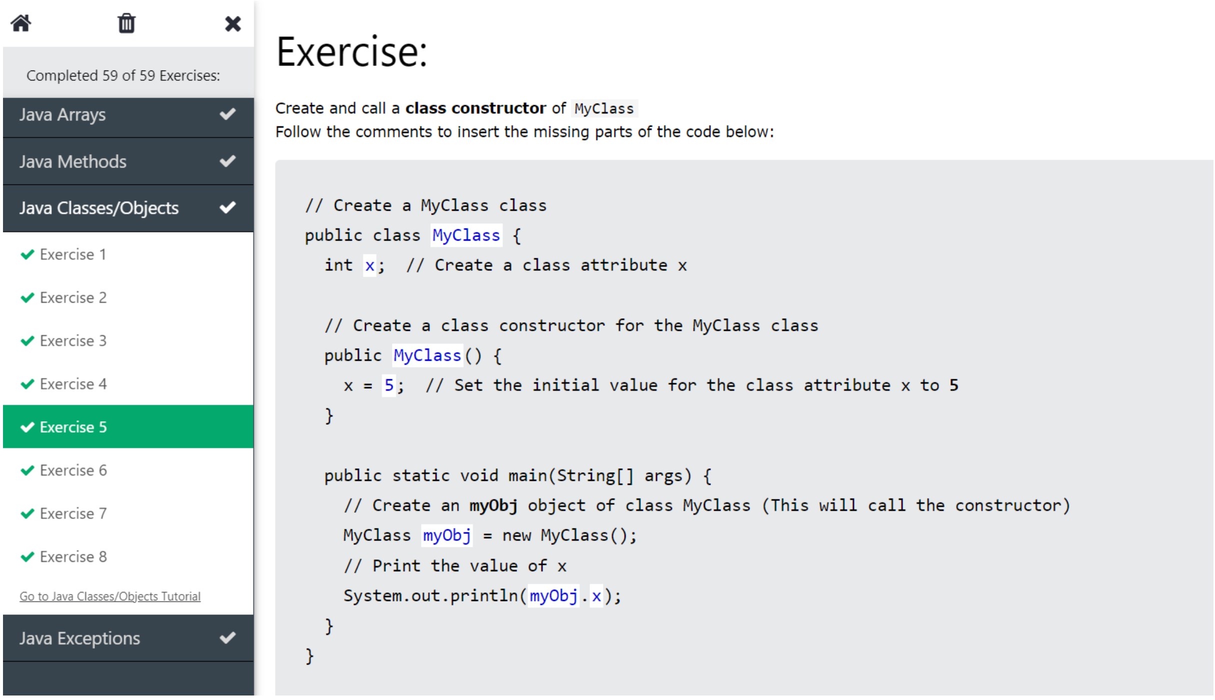
- Java Arrays ✓
- Java Methods ✓
- Java Classes/Objects ✓
- ✓ Exercise 1
- ✓ Exercise 2
- ✓ Exercise 3
- ✓ Exercise 4
- ✓ Exercise 5
- ✓ Exercise 6
- ✓ Exercise 7
- ✓ Exercise 8

Go to Java Classes/Objects Tutorial

Exercise:

Call `myMethod` on the object.

```
public class MyClass {
  public void myMethod() {
    System.out.println("Hello World");
  }

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    myObj.myMethod();
  }
}
```



Completed 59 of 59 Exercises:

- Java Arrays ✓
- Java Methods ✓
- Java Classes/Objects ✓
- ✓ Exercise 1
- ✓ Exercise 2
- ✓ Exercise 3
- ✓ Exercise 4
- ✓ Exercise 5
- ✓ Exercise 6
- ✓ Exercise 7
- ✓ Exercise 8

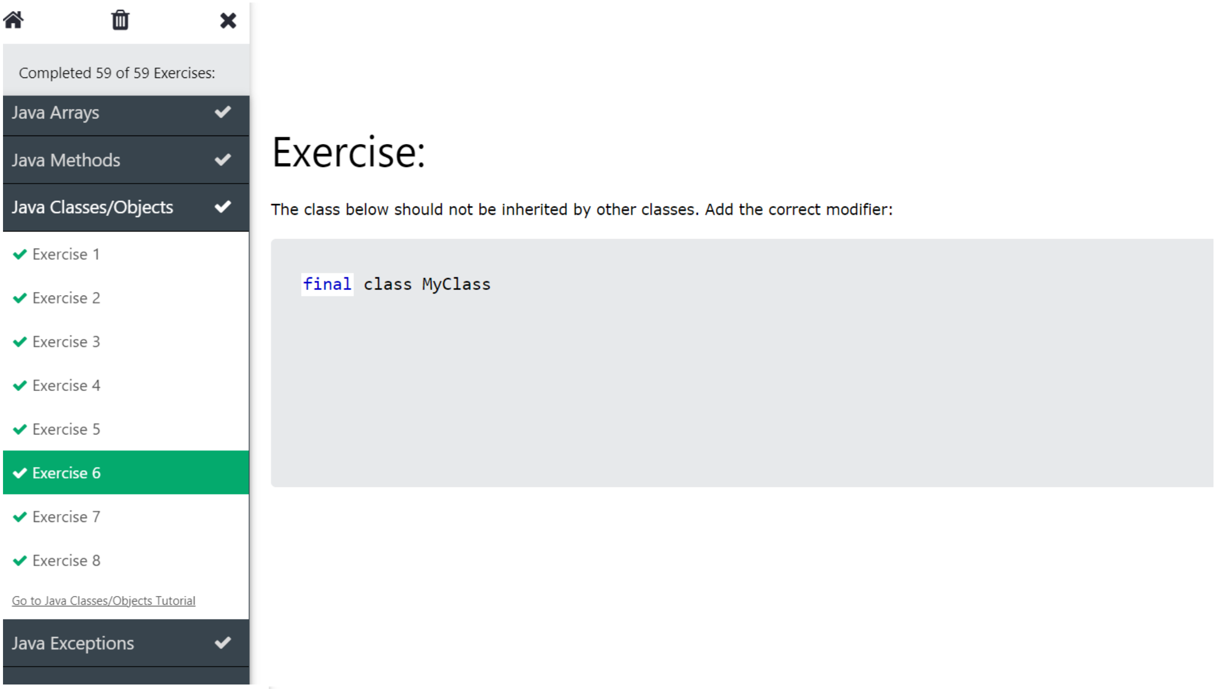
Go to Java Classes/Objects Tutorial

- Java Exceptions ✓

Exercise:

Create and call a **class constructor** of `MyClass`
Follow the comments to insert the missing parts of the code below:

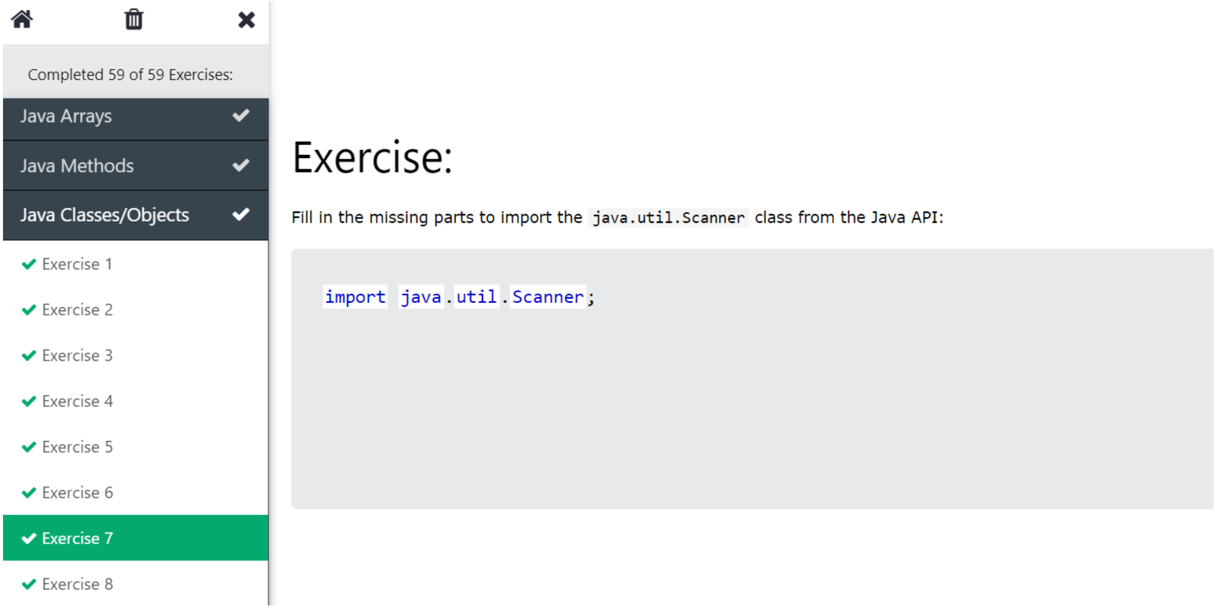
```
// Create a MyClass class
public class MyClass {
  int x;  // Create a class attribute x

  // Create a class constructor for the MyClass class
  public MyClass() {
    x = 5;  // Set the initial value for the class attribute x to 5
  }

  public static void main(String[] args) {
    // Create an myObj object of class MyClass (This will call the constructor)
    MyClass myObj = new MyClass();
    // Print the value of x
    System.out.println(myObj.x);
  }
}
```



Exercise:

The class below should not be inherited by other classes. Add the correct modifier:

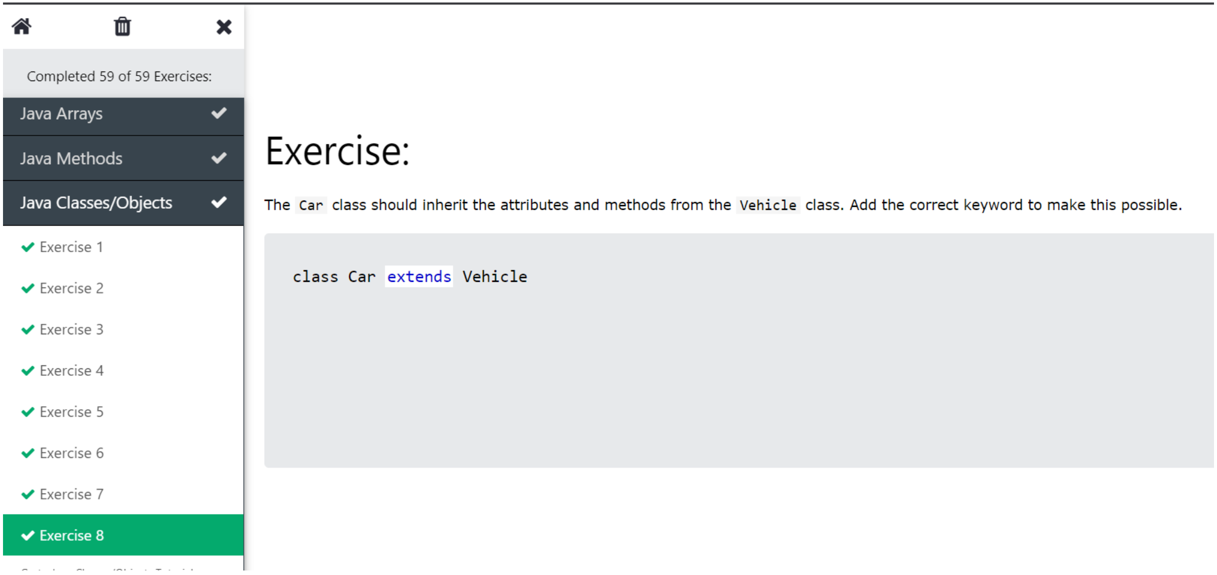
```
final class MyClass
```



Exercise:

Fill in the missing parts to import the `java.util.Scanner` class from the Java API:

```
import java.util.Scanner;
```

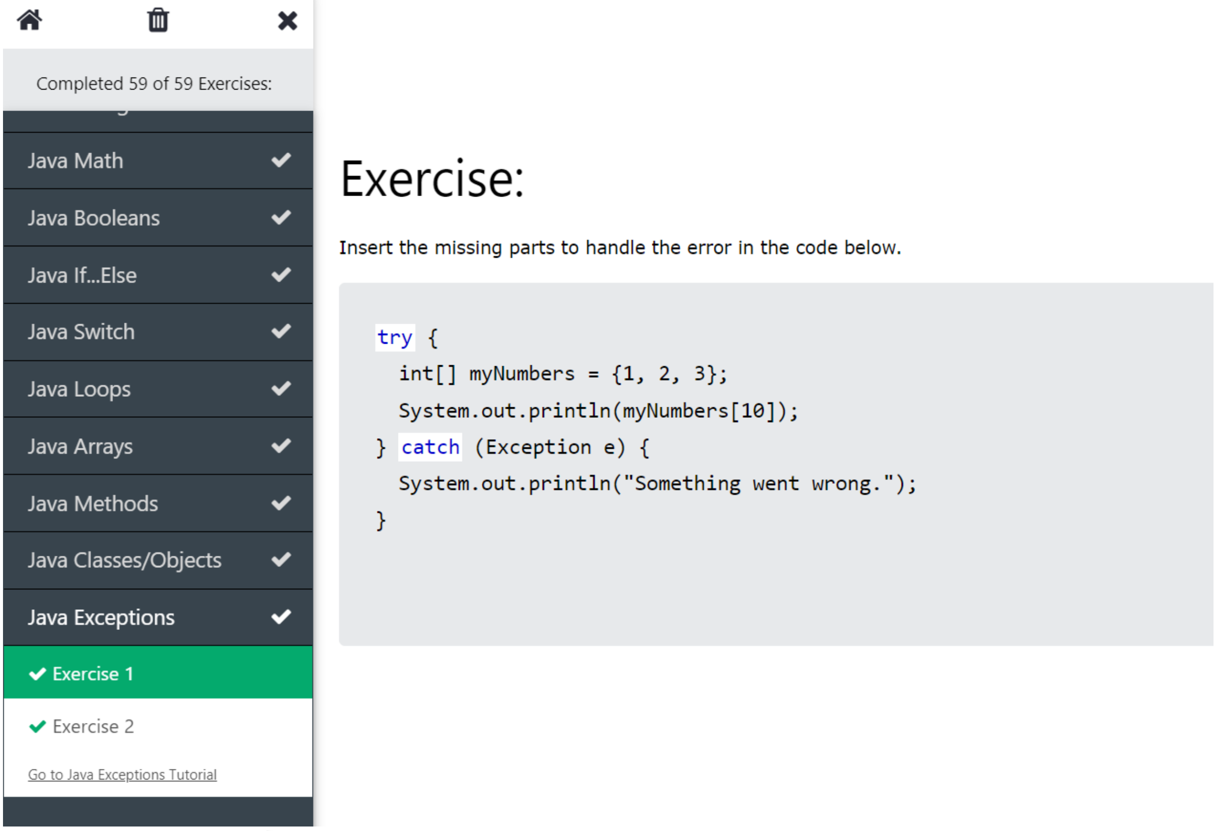


Exercise:

The `Car` class should inherit the attributes and methods from the `Vehicle` class. Add the correct keyword to make this possible.

```
class Car extends Vehicle
```

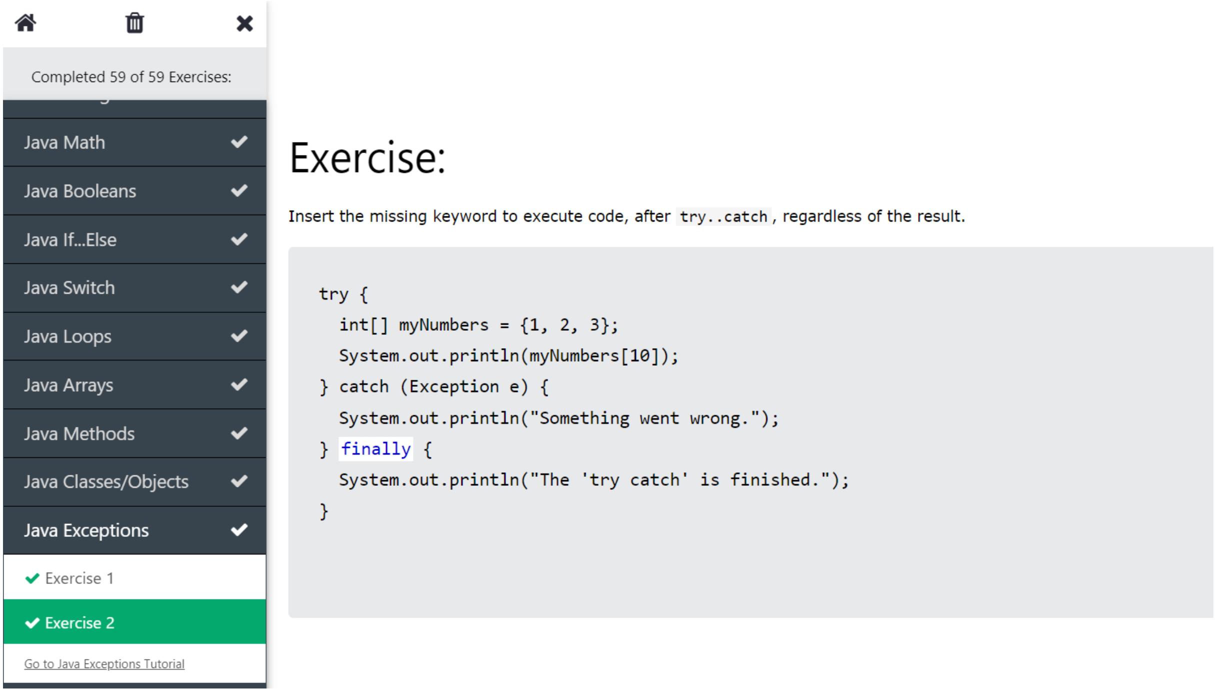
Exo sur les exception :



Exercise:

Insert the missing parts to handle the error in the code below.

```
try {
  int[] myNumbers = {1, 2, 3};
  System.out.println(myNumbers[10]);
} catch (Exception e) {
  System.out.println("Something went wrong.");
}
```



Exercise:

Insert the missing keyword to execute code, after `try..catch`, regardless of the result.

```
try {
  int[] myNumbers = {1, 2, 3};
  System.out.println(myNumbers[10]);
} catch (Exception e) {
  System.out.println("Something went wrong.");
} finally {
  System.out.println("The 'try catch' is finished.");
}
```