Method References

By Hatef Palizgar

Task 1

Create a class named "`Person`" that has two private fields, "`name`" and "`age`". Add a constructor that initializes these fields and getter methods for both fields. Then, create a functional interface named "`AgeChecker`" that has a single method named "`isAdult`" that takes a "`Person`" object as a parameter and returns a boolean indicating whether the person is an adult (age 18 or above).

Task 2 (This task requires Streams API)

Create a list of "`Person`" objects and populate it with at least five persons. Then, use a method reference to create an instance of the "`AgeChecker`" functional interface that checks if a person is an adult. Finally, use the "`filter`" method of the list to filter out all persons who are not adults, using the "`isAdult`" method reference you just created.

Task 3

Create a functional interface named "`StringConverter`" that has a single method named "`convert`" that takes a String as a parameter and returns an `integer`. Then, create a static method named "`countVowels`" in the "`Person`" class that takes a String as a parameter and returns the number of vowels in the string. Finally, use a method reference to create an instance of the "`StringConverter`" functional interface that converts a string to the number of vowels in it.

Hint: your `main` method should be like below:

```
public class Main {
    public static void main(String[] args) {
        StringConverter converter = Person::countVowels;
        int numVowels = converter.convert("Hello, world!");

        System.out.println(numVowels);
    }
}
```

Console Output:

```
3
```

Task 4

Create a class named "Calculator" that has two private fields, "num1" and "num2". Add a constructor that initializes these fields and getter methods for both fields. Then, create a functional interface named "BinaryOperator" that has a single method named "apply" that takes two integers as parameters and returns an integer. Next, create static methods in the "Calculator" class for each of the basic arithmetic operations (addition, subtraction, multiplication, and division) that take two integers as parameters and return the result of applying the respective operation.

Finally, inside main() method, create an instance of Calculator object with 10 and 5 as “num1” and “num2”, then use method references to create instances of the "BinaryOperator" functional interface that represent each of the basic arithmetic operations, call them "adder", "subtractor", "multiplier" and "divider".

Use the "apply" method of each of these "BinaryOperator" instances and pass them "num1" and "num2".

Print the result of each operation to the console.

Console Output:

```
Sum: 15
Difference: 5
Product: 50
Quotient: 2
```

Task 5

Create a class named "`StringUtils`" with a static method named "`reverse`" that takes a string as a parameter and returns the reverse of that string. Then, create a functional interface named "`StringFunction`" that has a single method named "`apply`" that takes a string as a parameter and returns a string. Finally, use a method reference to create an instance of the "`StringFunction`" functional interface that applies the "`reverse`" method of the "`StringUtils`" class to the input string `hello world`.

Console Output:

```
dlrow olleh
```

Task 6

Create a class named "`Person`" with private fields for "`name`" and "`age`", a constructor that initializes these fields, and getter methods for both fields. Then, inside `Person` class, create a static method named "`compareByAge`" that takes two "`Person`" objects as parameters and returns the difference in age between the two people. Finally, in your `main()` method create two `Person` instances `Person("Alice", 25)` and `Person("Bob", 30)`. Use a method reference to create an instance of the "`Comparator`" functional interface that compares "`Person`" objects by age using the "`compareByAge`" method; call this method reference `ageComparator`.

Calculate the age difference of `Alice` and `Bob` using the `ageComparator` and display it to the console.

Console Output:

```
Age difference: -5
```

Task 7

Create a class named "`Shape`" with private fields for "`name`", "`width`" and "`height`", a constructor that initializes only the "`name`" field, and a method named "`getArea`" that returns the area of the shape. Then, create subclasses of "`Shape`" for "`Rectangle`" and "`Circle`" that override the "`getArea`" method to compute the area of the respective shapes.

In your `main()` method create two "`Shape`" instances `Rectangle("Rectangle", 5.0, 3.0)` and `Circle("Circle", 2.5)`. Then use method references to create instances of the "`Function`" functional interface that compute the area of a "`Rectangle`" and a "`Circle`"; call them "`rectangleArea`" and "`CircleArea`" respectively. Use these two instances to print out the name and area of each shape instance to the console.

Console Output:

```
Rectangle area: 15.0
Circle area: 19.634954084936208
```

Task 8

Create a functional interface named "`StringProcessor`" that has a single method named "`process`" that takes a string as a parameter and returns a string. Then, create a class named "`StringUtils`" with static methods for reversing and capitalizing a string. Finally, use method references to create instances of the "`StringProcessor`" functional interface that correspond to the reverse and capitalize methods of the "`StringUtils`" class.

In your `main`() method try to print out to the console 3 items:

1. Original input string: “`Hello, World!`”
2. Reversed string
3. Capitalized string

Console Output:

```
Original String: Hello, World!
Reversed String: !dlroW ,olleH
Capitalized String: HELLO, WORLD!
```

TASK 9

Create a class named "Person" with a private field for "name" and a constructor that initializes this field. Then, create an instance method named "greet" that takes a string as a parameter and returns a greeting message with the person's name. Finally, create an instance of the "Person" class and use method references to create an instance of the "Function" functional interface that calls the "greet" method of the person instance.

The greet method should look like below:

```
public String greet(String message) {
    return "Hi, " + message + "! My name is " + name + ".";
}
```

Console Output:

```
Hi, how are you! My name is John.
```

Task 10

Create a class named "`Car`" with a private field for "`make`" and "`model`" and a constructor that initializes this field. Then, create a functional interface named "`CarFactory`" that has a single method named "`create`" that takes two string parameters for the "`make`" and "`model`" of the car and returns a new instance of the "`Car`" class with the given `make` and `model`. Finally, use method references to create an instance of the "`CarFactory`" functional interface that corresponds to the "`Car`" constructor. In your `main()` method use this "`CarFactory`" instance to create a "`Car`" object and print it out to the console.

Console Output:

```
Created car with make: Toyota, model: Camry
```