Interface & Lambda Expressions

COMP2026

PROBLEM SOLVING USING OBJECT ORIENTED PROGRAMMING

Interface

Interface

- **not a class**
- a set of requirements, in the form of a group of related **methods**
- To apply the interface, create a class with the **implements** keyword in the class declaration
- The class has to provide the method bodies of all abstract methods specified by the interface

Example

```
public interface Movable
{
   int DX = 5;
   int DY = 5;

   int getX();
   int getY();
   void setX(int x);
   void setY(int y);
   void moveLeft();
   void moveRight();

}
```

Implicitly: public static final constants

Implicitly: public abstract methods

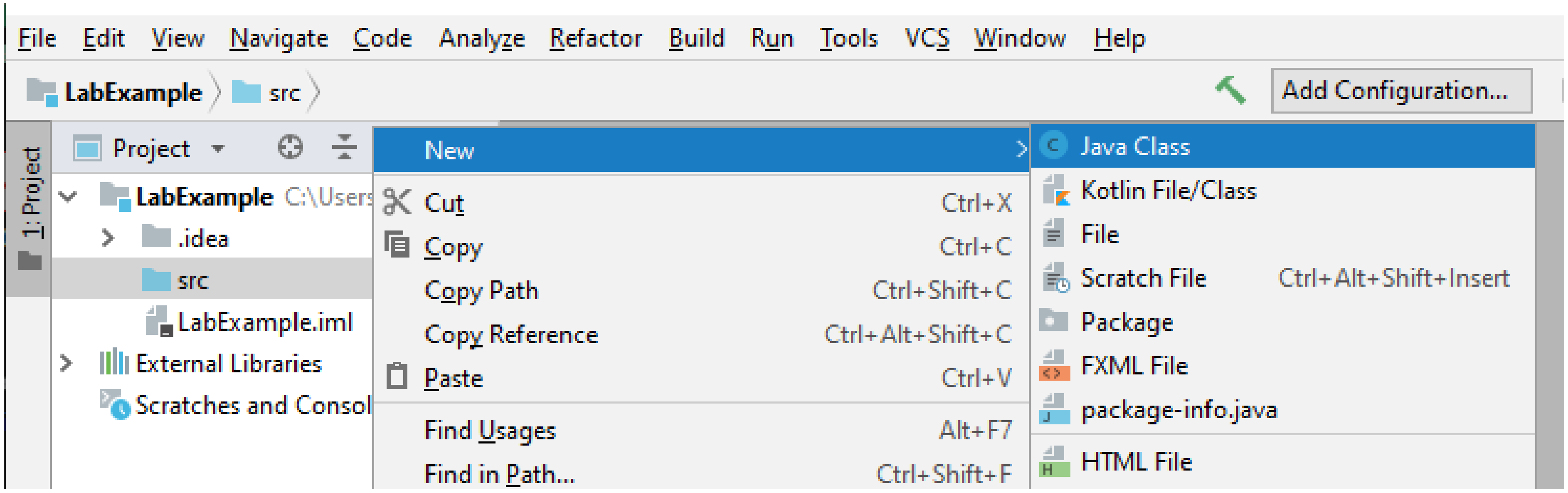
```
public class Point implements Movable
{
  private int x;
  private int y;

  public Point(int x, int y){
    this.x = x;
    this.y = y;
  }
  public int getX(){
     return this.x;
  }
  public int getY(){
     this.y;
  }
  public void setX(int x){
     this.x = x;
  }
  public void setY(int y){
     this.y = y;
  }
  public void moveLeft(){
     this.x = this.x - DX;
  }
  public void moveRight(){
     this.x = this.x + DX;
  }
  ...
}
```

Keyword: implements

Concrete implementation of the abstract methods

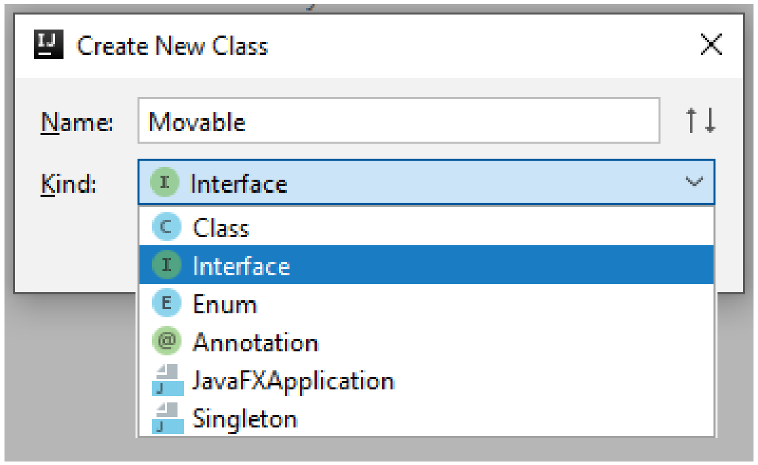
Creating Interface in IntelliJ

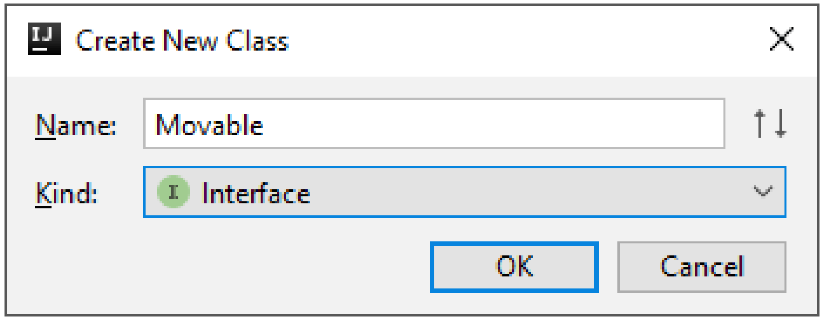
- In an IntelliJ project, Right click the **src** folder
- Then choose **New > Java Class**



Creating Interface in IntelliJ

- Type in the interface name and select **Interface** in the Kind textbox, then click **OK**





Lambda Expressions

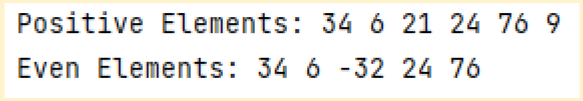
Consider the following single method interface…

```java
public interface NumChecker {
    public boolean check(int n);
}
```

```java
public class PositiveChecker implements NumChecker{
    public boolean check(int n) {
        return n > 0;
    }
}
```

```java
public class EvenChecker implements NumChecker{
    public boolean check(int n) {
        return n %2 == 0;
    }
}
```

```java
public class MyProg1 {
    public static void main(String[] args){
        new MyProg1().runApp();
    }

    //Print all the elements in the given integer array that pass the check
    public void printElements(int[] a, NumChecker c){
        for (int i = 0; i < a.length; i++){
            if(c.check(a[i])){
                System.out.print(a[i] + " ");
            }
        }
        System.out.println();
    }

    public void runApp(){
        int[] intAry = {34, 6, 21, -1, -32, 24, -97, 76, 9};

        System.out.print("Positive Elements: ");
        NumChecker pc = new PositiveChecker(); //create PositiveChecker object
        printElements(intAry, pc); //pass the object to the method

        System.out.print("Even Elements: ");
        NumChecker ec = new EvenChecker(); //create EvenChecker object
        printElements(intAry, ec); //pass the object to the method
    }
}
```

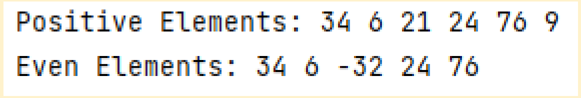


```
Positive Elements: 34 6 21 24 76 9
Even Elements: 34 6 -32 24 76
```

Using anonymous class

```java
public interface NumChecker {
    public boolean check(int n);
}
```

```java
public class MyProg2 {
    public static void main(String[] args){ … }

    //Print all the elements in the given integer array that pass the check
    public void printElements(int[] a, NumChecker c){ … }

    public void runApp(){
        int[] intAry = {34, 6, 21, -1, -32, 24, -97, 76, 9};

        System.out.print("Positive Elements: ");
        printElements(intAry, new NumChecker(){ //Anonymous class
            public boolean check(int n){
                return n > 0;
            }});
        System.out.print("Even Elements: ");
        printElements(intAry, new NumChecker(){ //Anonymous class
            public boolean check(int n){
                return n % 2 == 0;
            }});
    }
}
```

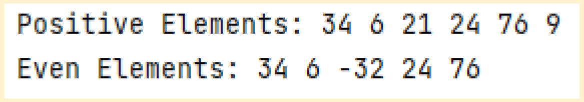


```
Positive Elements: 34 6 21 24 76 9
Even Elements: 34 6 -32 24 76
```

Using lambda expression

```java
public interface NumChecker {
    public boolean check(int n);
}
```

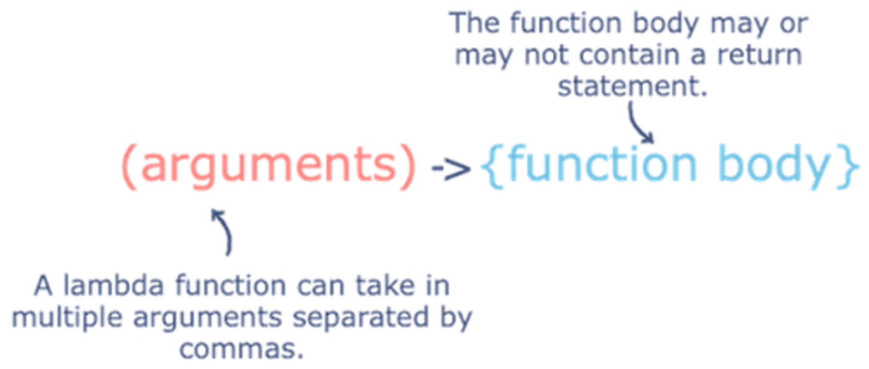
```java
public class MyProg3 {
    public static void main(String[] args){ … }

    //Print all the elements in the given integer array that pass the check
    public void printElements(int[] a, NumChecker c){ … }

    public void runApp(){
        int[] intAry = {34, 6, 21, -1, -32, 24, -97, 76, 9};

        System.out.print("Positive Elements: ");
        printElements(intAry, (n)->{return n > 0;}); //Lambda expression
        System.out.print("Even Elements: ");
        printElements(intAry, (n)->{return n%2 == 0;}); //Lambda expression
    }
}
```



```
Positive Elements: 34 6 21 24 76 9
Even Elements: 34 6 -32 24 76
```

Lambda Expression

- A lambda expression is a short block of code which takes in parameters and returns a value.
- Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method.
- Syntax:


The function body may or may not contain a return statement.
(arguments) -> {function body}
A lambda function can take in multiple arguments separated by commas.

  - There can be zero or more arguments. If there is more than one argument, then they need to be enclosed inside the parenthesis.
  - If the function body consists of only one line, then curly braces are optional.
  - The function body may or may not contain a return statement.

Source: https://www.educative.io/edpresso/what-is-a-java-lambda-function

Part A
Discovery Exercises

Type your answer in **XXXXXXXX_lab13.docx**

Part B
Programming Exercises

Hints for Task 5

- To get today's date

```
LocalDate myDate = LocalDate.now();
```

- To add days to the date

```
myDate.plusDays(3); //return a LocalDate
                    //with 3 days after MyDate
```

Optional Exercises on Recursion

Writing recursive method

- Before we write a recursive method, the first thing we have to do is to **THINK recursively**
- How?
  1) Find the base case(s) to stop / end the method
  2) Reduce the problem into similar and smaller sub-problem
  3) Assume the method is done and use the method to solve the sub-problem inside the method itself

Example

- Write a recursive method that computes the sum of all numbers from 1 to n, where n is given as parameter.

```
//return the sum 1 + 2 + 3 + … + n
int sum(int n)
 {




 }
```

Example

- Write a recursive method that computes the sum of all numbers from 1 to n, where n is given as parameter.

```
//return the sum 1 + 2 + 3 + … + n
int sum(int n)
 {
    if (n == 1)
      return 1;


 }
```

Think:

1) Add base case to end the method
   when n = 1, the method should return 1

Example

- Write a recursive method that computes the sum of all numbers from 1 to n, where n is given as parameter.

```
//return the sum 1 + 2 + 3 + … + n
int sum(int n)
 {
    if (n == 1)
      return 1;
   reutn n + [(n-1) + … + 1];
 }
```

Think:

2) Reduce the problem:
   sum of n = n + sum of (n – 1)

Example

- Write a recursive method that computes the sum of all numbers from 1 to n, where n is given as parameter.

```
//return the sum 1 + 2 + 3 + … + n
int sum(int n)
 {
    if (n == 1)
       return 1;
   reutn n + sum(n-1);
 }
```

Think:
3) Assume this sum() method is done and use sum(n-1) in our method

Lab Ex. Submission

- Submit the following to Moodle
  - ***XXXXXXXX*_ lab13.docx**
  - ***XXXXXXXX*_lab13.zip**

*Replace “***XXXXXXXX***” with your **student ID**

Deadline: Next Wednesday noon

References

- Dean, J., & Dean, R. (2008). Introduction to programming with Java: A problem solving approach. Boston: McGraw-Hill.
- Forouzan, B. A., & Gilberg, R. F. (2007). Computer science: A structured programming approach using C (3rd ed.). Boston, MA: Thomson Course Technology.
- Gaddis, T. (2016). Starting out with Java (6th ed.). Pearson.
- Liang, Y. D. (2013). *Introduction to Java programming: Comprehensive version*. (8th ed.). Pearson.
- Schildt, H. (2006). *Java a beginner's guide*. New York: McGraw Hill.
- Wu, C. T. (2010). *An introduction to object-oriented programming with Java*. Boston: McGraw Hill Higher Education
- Xavier, C. (2011). Java programming: A practical approach. New Delhi: Tata McGraw Hill.
- Zakhour, S., Kannan, S., & Gallardo, R. (2013). The Java tutorial: A short course on the basics (5th ed.).
- yet another insignificant Programming Notes. (n.d.). Retrieved from https://www3.ntu.edu.sg/home/ehchua/programming
- Edpresso Team. (2019, July 1). What is a Java lambda function? Educative: Interactive Courses for Software Developers. https://www.educative.io/edpresso/what-is-a-java-lambda-function