

I. Bài tập thực hành

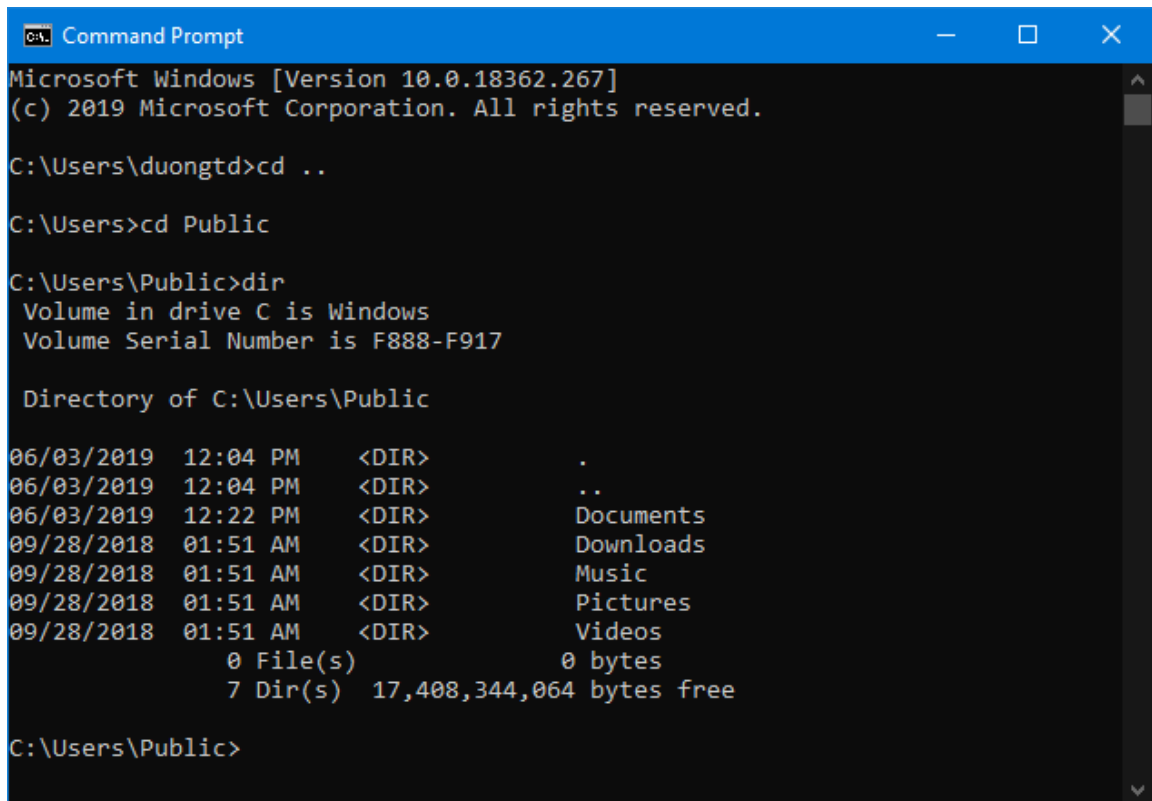
Tuần 1. Cài đặt môi trường

Viết chương trình in ra dòng chữ “Hello World”, làm trực tiếp trên hệ thống hoặc làm trên máy tính tại lab hoặc máy tính cá nhân.

Nếu triển khai trên máy tính tại lab, bạn chỉ cần làm theo từ **Bước 4** trong phần **Triển khai trên máy tính cá nhân**.

Triển khai trên máy tính cá nhân:

1. Tải và cài đặt Oracle JDK8 tại [đây](#). Lưu ý, ở đây sử dụng Oracle JDK (ngoài ra có thể tìm hiểu thêm OpenJDK), Java SE, phiên bản 1.8 (đã có nhiều JDK version mới hơn, tuy nhiên để thống nhất sẽ chọn JDK8).
2. Làm quen dòng lệnh trên Windows (trên Linux tương tự, chỉ 1 vài khác biệt nhỏ), các lệnh cơ bản (cd, dir/ls, copy/cp, etc):
 - Mở cmd bằng cách tìm kiếm “cmd”, hoặc dùng **Windows + R** -> gõ “cmd” -> Enter
 - Lệnh cd (change directory): câu lệnh này dùng để thay đổi vị trí thư mục hiện tại – di chuyển đến vị trí thư mục khác. Một số cách khác nhau khi sử dụng lệnh cd là:
 - cd . : đứng nguyên ở thư mục hiện tại
 - cd .. : di chuyển đến thư mục cha của thư mục hiện tại
 - cd / : di chuyển đến thư mục gốc của ổ đĩa hiện tại (ví dụ: C:\ hoặc D:\,...)
 - cd <tên thư mục con> : di chuyển đến thư mục con bên trong thư mục hiện tại
 - cd <đường dẫn đến thư mục> : di chuyển đến thư mục với đường dẫn là đường dẫn cho trước.
 - Lệnh dir (directory), (trên Linux tương ứng là ls): Hiển thị danh sách các tập tin và thư mục trong thư mục hiện tại.



```
C:\> Command Prompt
Microsoft Windows [Version 10.0.18362.267]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\duongtd>cd ..

C:\Users>cd Public

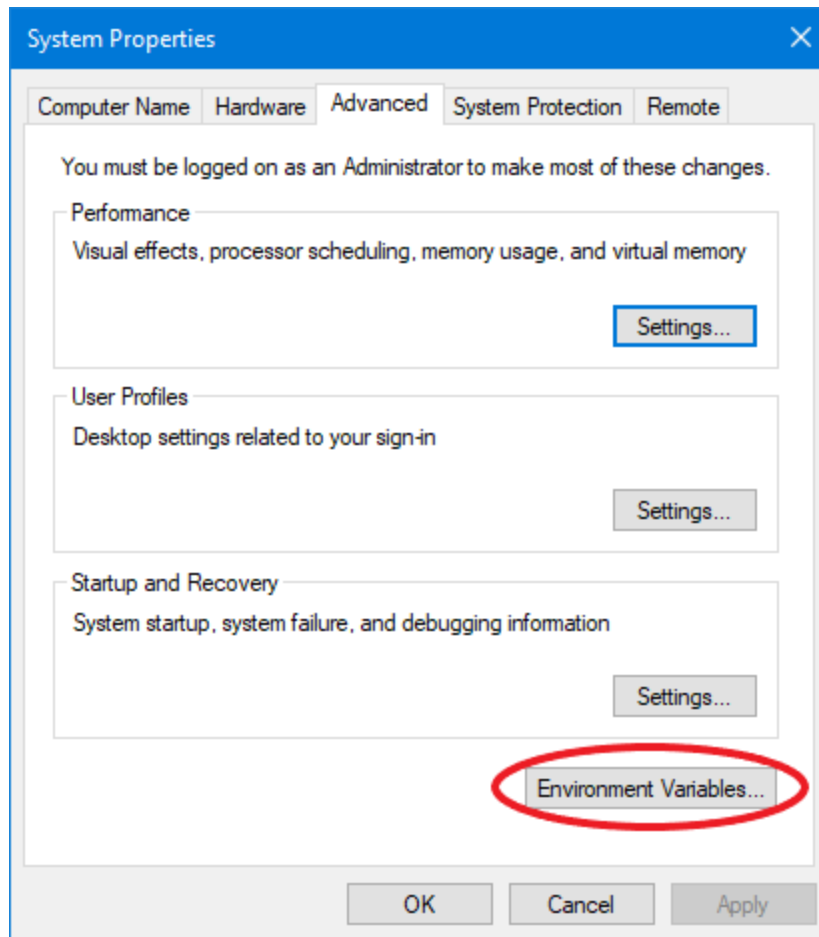
C:\Users\Public>dir
Volume in drive C is Windows
Volume Serial Number is F888-F917

Directory of C:\Users\Public

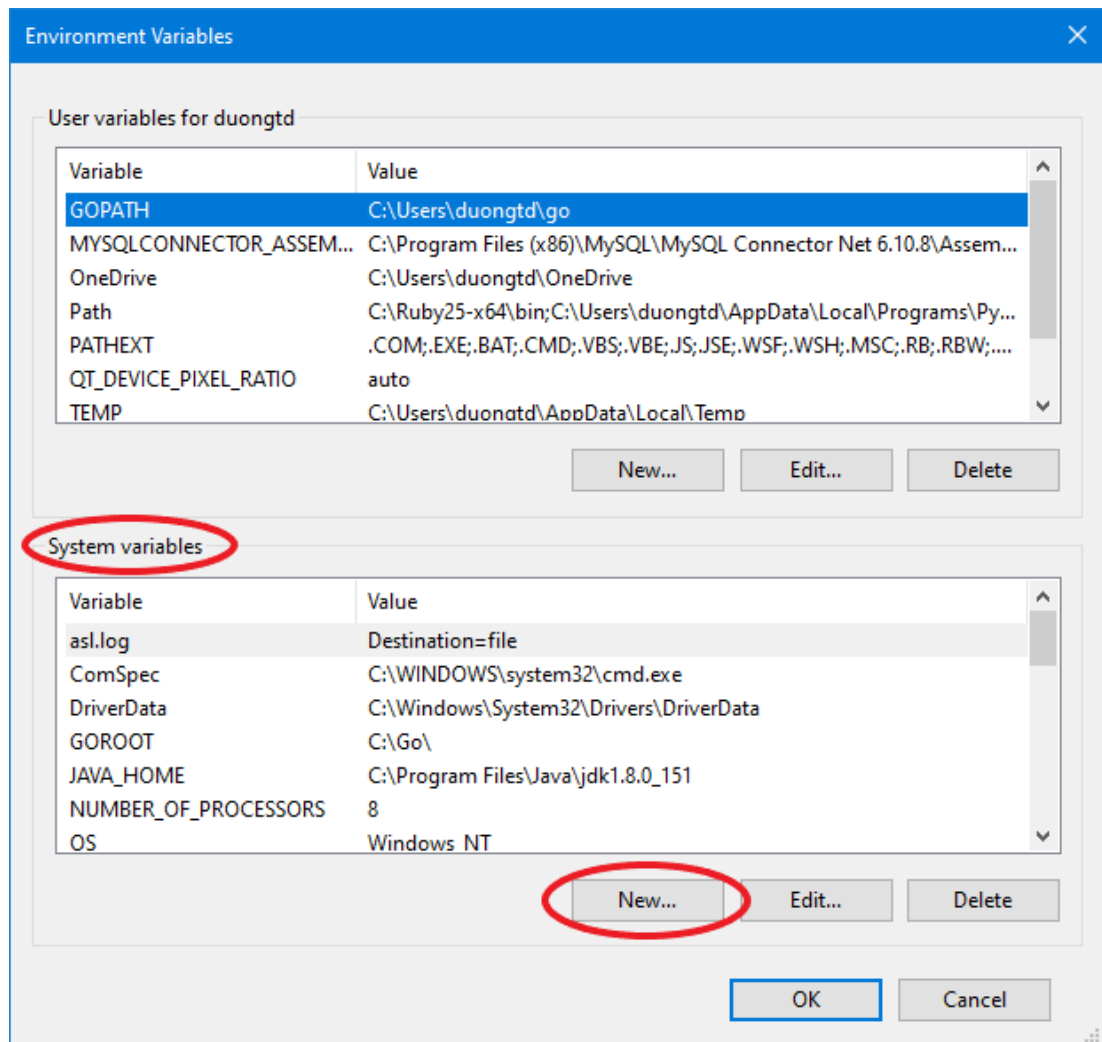
06/03/2019  12:04 PM    <DIR>          .
06/03/2019  12:04 PM    <DIR>          ..
06/03/2019  12:22 PM    <DIR>          Documents
09/28/2018  01:51 AM    <DIR>          Downloads
09/28/2018  01:51 AM    <DIR>          Music
09/28/2018  01:51 AM    <DIR>          Pictures
09/28/2018  01:51 AM    <DIR>          Videos
               0 File(s)              0 bytes
               7 Dir(s)  17,408,344,064 bytes free

C:\Users\Public>
```

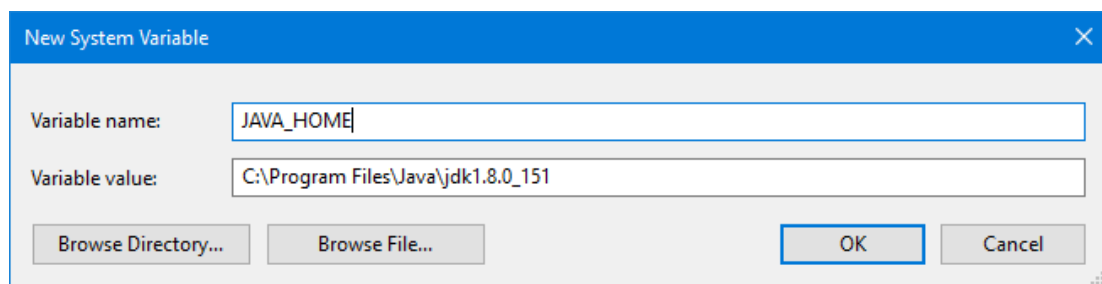
3. Thêm Java vào biến môi trường Path, test thử lệnh java, javac
 - Sử dụng Windows search (Windows + S) tìm kiếm “Edit the system environment variable”. Trong hộp thoại “System Properties” hiện lên, chọn “Environment Variables” như hình.



Trong hộp thoại Environment Variables, vào phần System variables và chọn New.

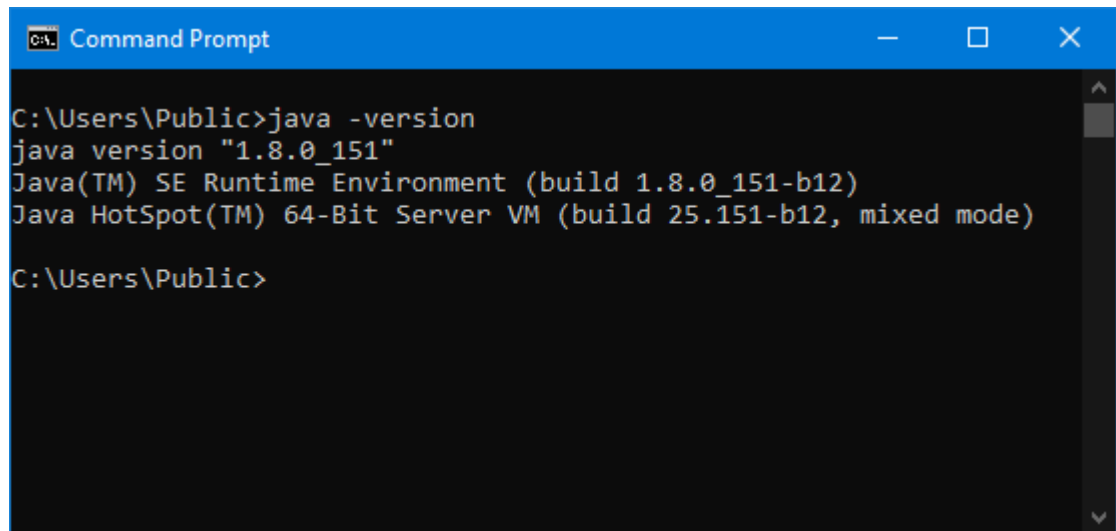


Trong hộp thoại New System Variable, điền vào ô **Variable name** là **JAVA_HOME**, còn trong ô **Variable value** nhấn vào nút **Browser Directory** và trở tới đường dẫn cài đặt JDK. Nhấn OK để hoàn tất việc đặt tên biến môi trường.



Tiếp theo, cũng trong phần System variables, tiến hành sửa đổi biến môi trường **Path** như sau: Kích chuột vào dòng **Path** và chọn **Edit**. Hộp thoại **Edit environment variable** xuất hiện, chúng ta nhấn chuột vào nút **New** và điền vào dòng sau: **%JAVA_HOME%\bin,;**, nhấn OK để kết thúc.

- Kiểm tra bằng cách mở cmd, nhập vào dòng: **java -version**. Nếu thông tin hiển thị ra là version Java tương tự như hình dưới là thành công



```

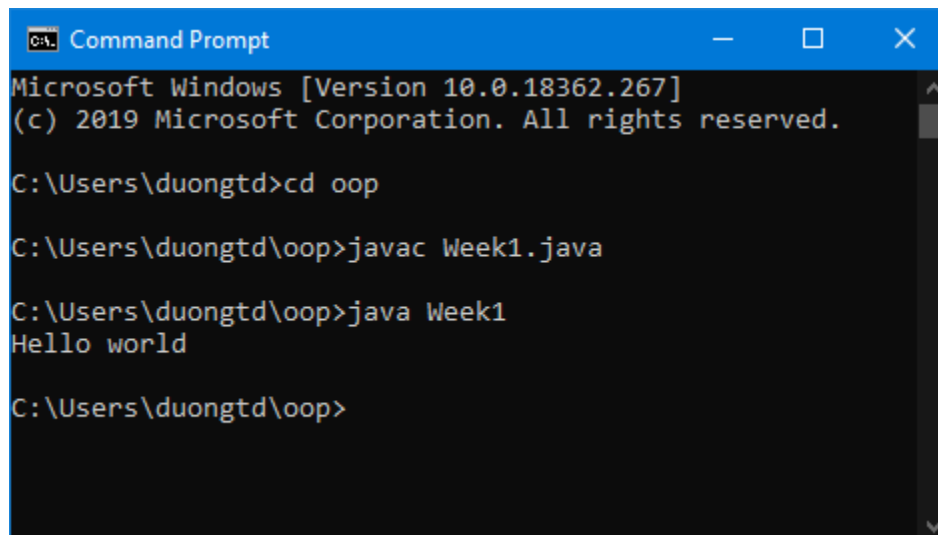
C:\Users\Public>java -version
java version "1.8.0_151"
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)

C:\Users\Public>

```

4. Sau khi đã hoàn tất việc cài đặt Java, hãy viết chương trình đơn giản in ra dòng “Hello World”, biên dịch và thực thi sử dụng command. Giả sử code được lưu trong file Week1.java, gọi ý các bước làm như sau:

- Sử dụng lệnh “cd” để đến thư mục chứa file Week1.java.
- Thực thi command “javac Week1.java” để biên dịch mã nguồn, nếu không có lỗi in ra là thành công.
- Thực thi command “java Week1”, nếu thành công sẽ in ra dòng chữ “Hello World”.



```

Microsoft Windows [Version 10.0.18362.267]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\duongtd>cd oop

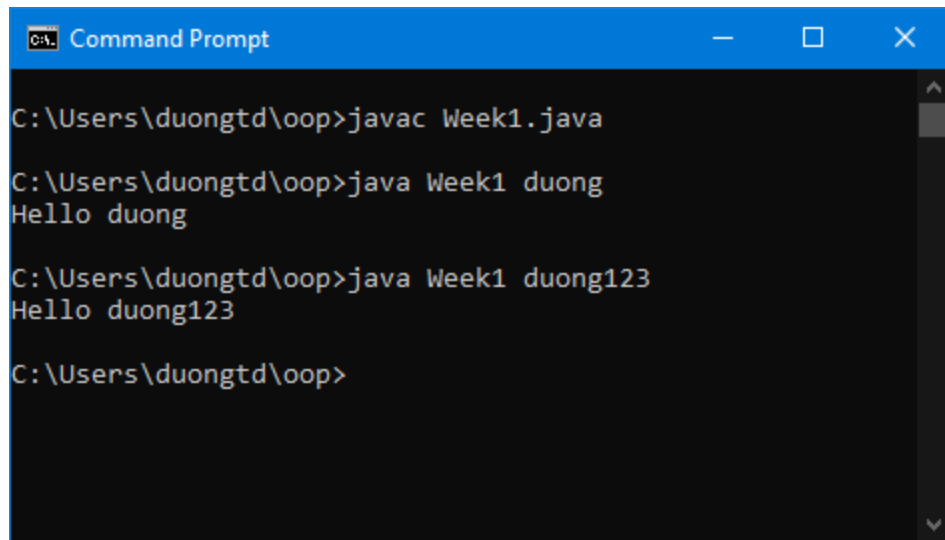
C:\Users\duongtd\oop>javac Week1.java

C:\Users\duongtd\oop>java Week1
Hello world

C:\Users\duongtd\oop>

```

5. Thêm tham số dòng lệnh, in ra Hello student_name, trong đó student_name là tên được truyền vào bằng tham số dòng lệnh.



```
C:\Users\duongtd\oop>javac Week1.java  
C:\Users\duongtd\oop>java Week1 duong  
Hello duong  
C:\Users\duongtd\oop>java Week1 duong123  
Hello duong123  
C:\Users\duongtd\oop>
```

The image shows a Windows Command Prompt window with a blue title bar labeled "C:\ Command Prompt". The window contains four lines of text representing a Java compilation and execution process. The first line shows the compilation of "Week1.java" using the "javac" command. The second line shows the execution of "Week1.java" with the argument "duong", resulting in the output "Hello duong". The third line shows the execution of "Week1.java" with the argument "duong123", resulting in the output "Hello duong123". The fourth line shows the prompt "C:\Users\duongtd\oop>" without any further input or output. A vertical scrollbar is visible on the right side of the command prompt window.

Tuần 2. Setter/Getter, Constructor

Từ tuần 2, khuyến khích sử dụng IDE (ví dụ IntelliJ) để code thay vì code, biên dịch và thực thi trên command line. Các hướng dẫn trong đây đều mặc định với IDE IntelliJ.

Để thể hiện cho đối tượng con người, mã nguồn lớp Person dưới đây được cài đặt minh họa. Nó đơn giản bao gồm một thuộc tính name – là tên người; 2 phương thức khởi tạo, 1 không có tham số, và 1 có tham số là tên người; getter/setter cho thuộc tính name. Chi tiết thể hiện trong comment mã nguồn.

```
public class Person {
    String name; // thuộc tính name

    // phương thức khởi tạo không tham số
    public Person() {
    }

    // phương thức khởi tạo 1 tham số (name)
    public Person(String name) {
        this.name = name;
    }

    // getter cho thuộc tính name
    public String getName() {
        return name;
    }

    // setter cho thuộc tính name
    public void setName(String name) {
        this.name = name;
    }
}
```

Sau khi đã đọc hiểu ví dụ bên trên, hãy tự cài đặt lớp Student theo yêu cầu cụ thể dưới đây.

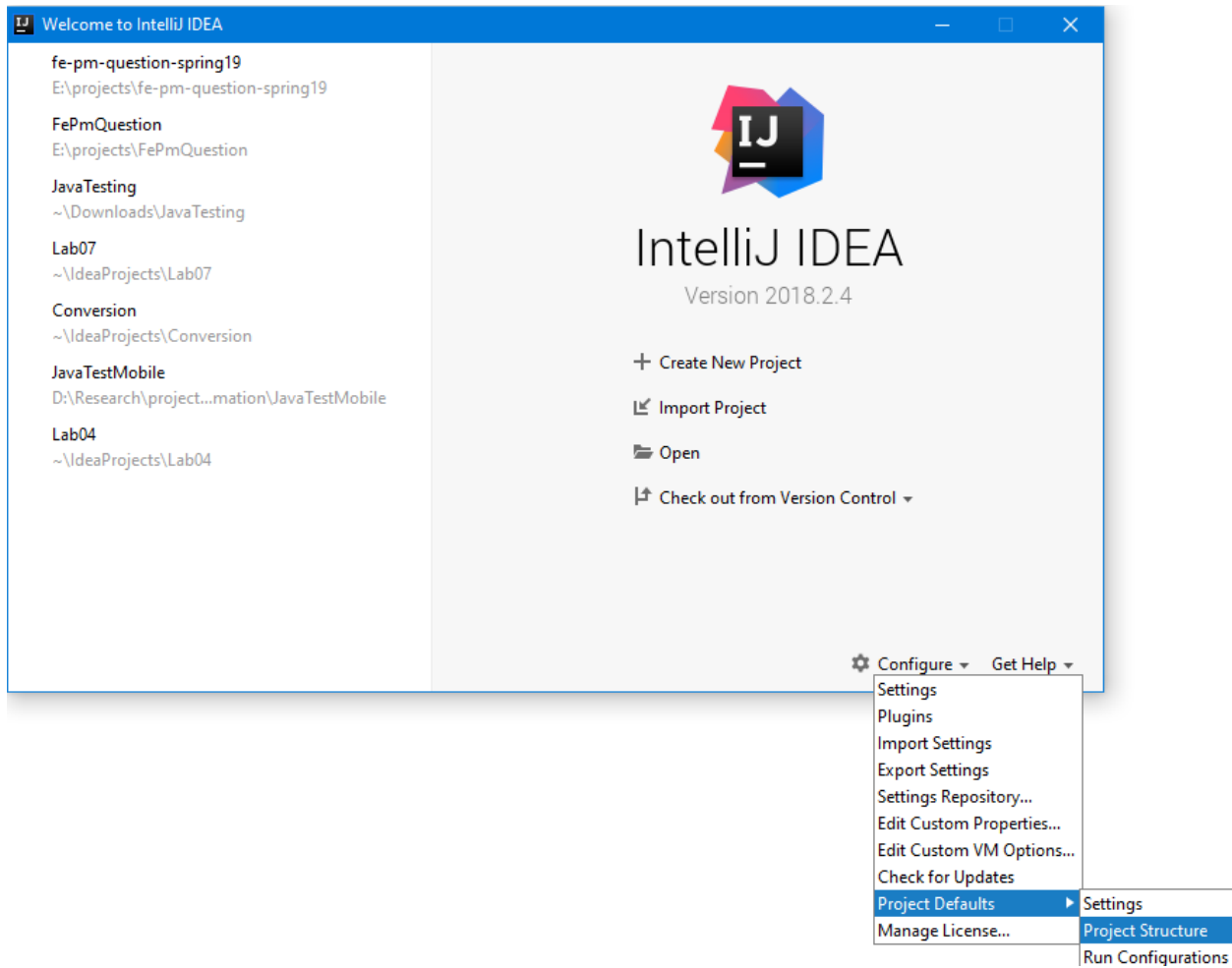
1. Tạo lớp **Student** với các thuộc tính kiểu String, mức truy nhập *private* gồm: **name** (tên sinh viên), **id** (mã số sinh viên), **group** (lớp học), **email** (địa chỉ email).
2. Thêm các phương thức get/set cho các thuộc tính (gọi là getter/setter). Ví dụ, với thuộc tính “name”, hai phương thức cần thêm gồm “**public String getName()**” và “**public void setName(String n)**”.
3. Tạo đối tượng Student có tên là *Nguyen Van An*, id là *17020001*, lớp *K62CC*, email *17020001@vnu.edu.vn*.
Thêm phương thức “**String getInfo()**” cho lớp Student. Phương thức này trả về tên, mã số SV, lớp, và email của sinh viên theo đúng định dạng sau:
Nguyen Van An – 17020001 – K62CC – 17020001@vnu.edu.vn
4. Thêm 3 phương thức khởi tạo cho lớp Student

- Phương thức khởi tạo không có tham số: **Student()**. Nếu khởi tạo bằng phương thức này, sinh viên được tạo ra sẽ có giá trị cho các thuộc tính như sau: name = "Student", id="000", group="K62CB", email = "uet@vnu.edu.vn"
 - Phương thức khởi tạo có tham số **Student(String name, String id, String email)**. Khởi tạo bằng phương thức này sẽ có sinh viên với các thuộc tính "name", "id", và "email" là các giá trị từ tham số, còn "group" có giá trị là "K62CB".
 - Phương thức khởi tạo sao chép **Student(Student s)**. Với phương thức này, đối tượng tạo ra sẽ có các thuộc tính với trị giống như của đối tượng s.
5. Tạo thêm lớp **StudentManagement** (tự tạo phương thức **main()** để kiểm tra các chức năng cài đặt bên trên). Trong lớp StudentMangement, viết một phương thức "**public static boolean sameGroup(Student s1, Student s2)**" để kiểm tra xem hai sinh viên s1 và s2 có cùng lớp hay không.
 6. Sửa lại lớp StudentManagement để lớp này có một thuộc tính students là array (không dùng List) chứa các đối tượng thuộc lớp Student (max. 100) có tên **students**. Viết phương thức **public void addStudent(Student newStudent)** để thêm mới một Student vào mảng.
 7. Viết phương thức "**public String studentsByGroup()**" cho lớp StudentManagement trả về chuỗi in danh sách sinh viên theo lớp tuân theo định dạng sau (lưu ý lớp sắp xếp theo thứ tự xuất hiện đầu tiên, sinh viên sắp xếp theo thứ tự thêm vào):
 K62CC
 Nguyen Van An - 17020001 - K62CC - 17020001@vnu.edu.vn
 Nguyen Van B - 17020002 - K62CC - 17020002@vnu.edu.vn
 K62CB
 Nguyen Van C - 17020003 - K62CB - 17020003@vnu.edu.vn
 Nguyen Van D - 17020004 - K62CB - 17020004@vnu.edu.vn
Thứ tự thêm sinh viên ở trên là An, B, C, D (thêm sinh viên bằng phương thức addStudent).
 8. Viết phương thức "**public void removeStudent(String id)**" cho lớp StudentManagement để xóa sinh viên với mã số là id ra khỏi danh sách.

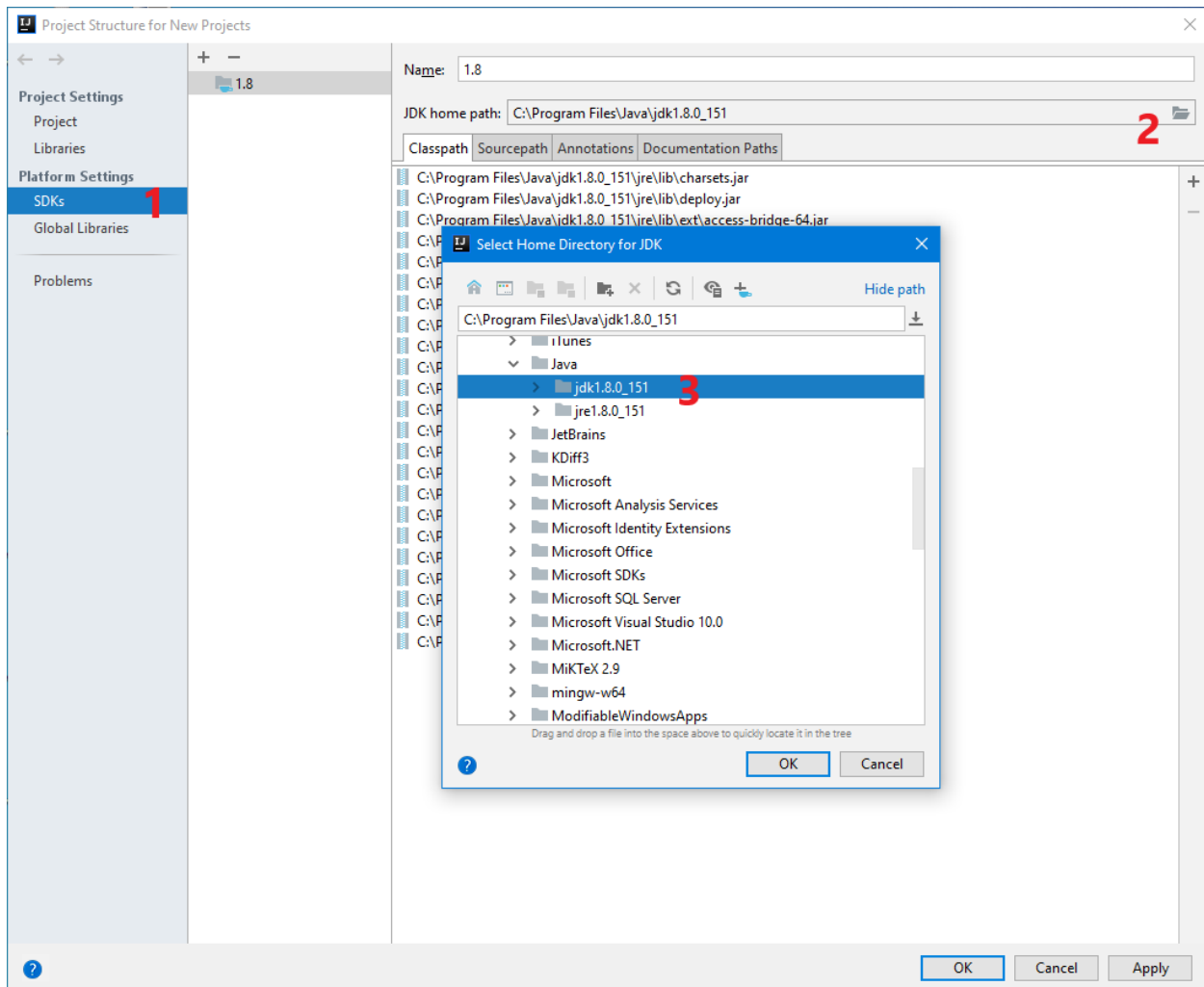
Hướng dẫn:

Sau khi đã tải và cài đặt IntelliJ, nếu chạy lần đầu, cần config đường dẫn tới JDK đã cài đặt ở tuần 1:

6. Click *Configure -> Project Defaults -> Project Structure*

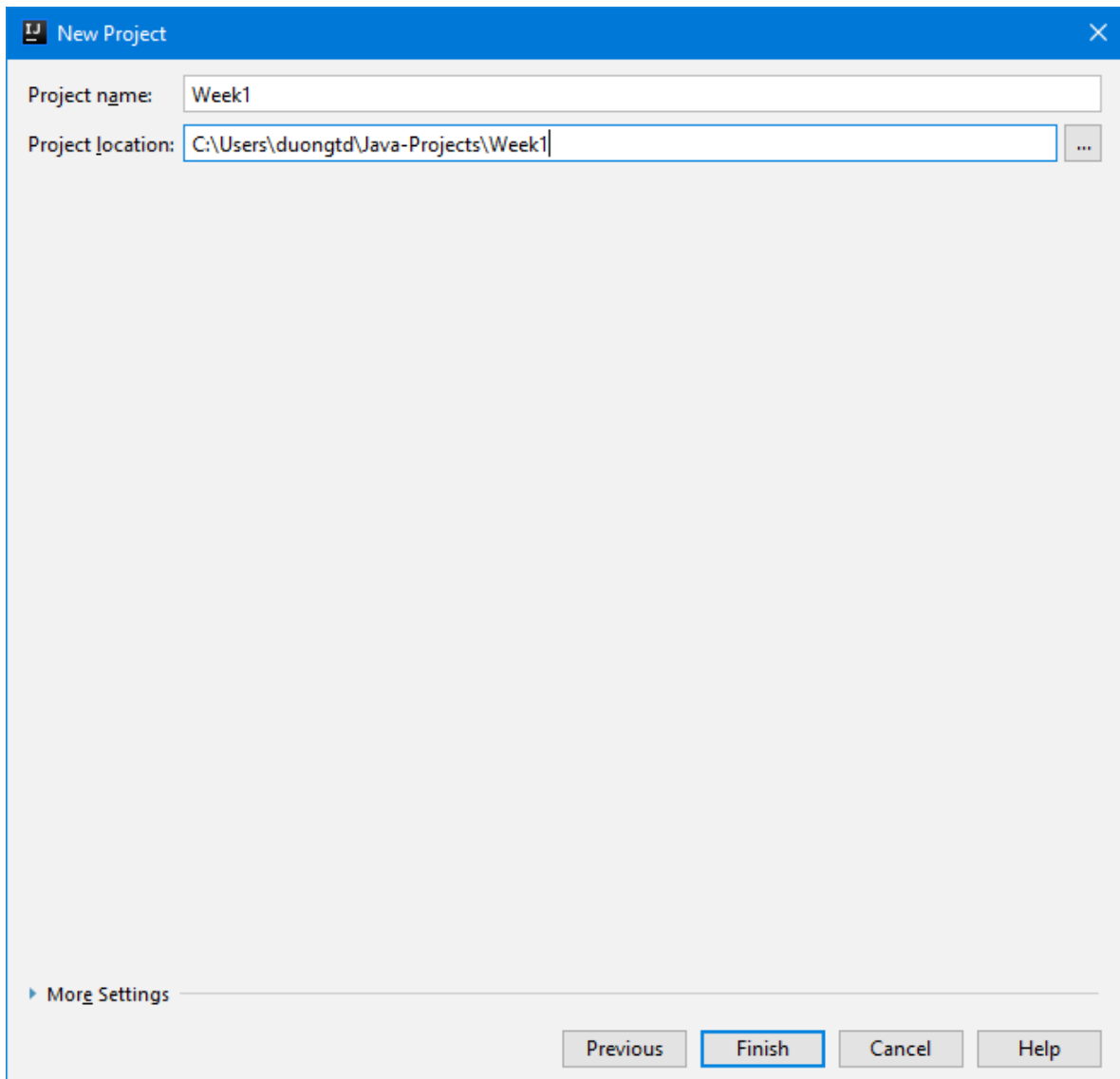


7. Cửa sổ mới hiện lên, bên trái chọn *SDKs*; bên phải click để chọn đường dẫn tới *JDK home path*. Sau khi chọn thành công click *OK* để ghi nhận.

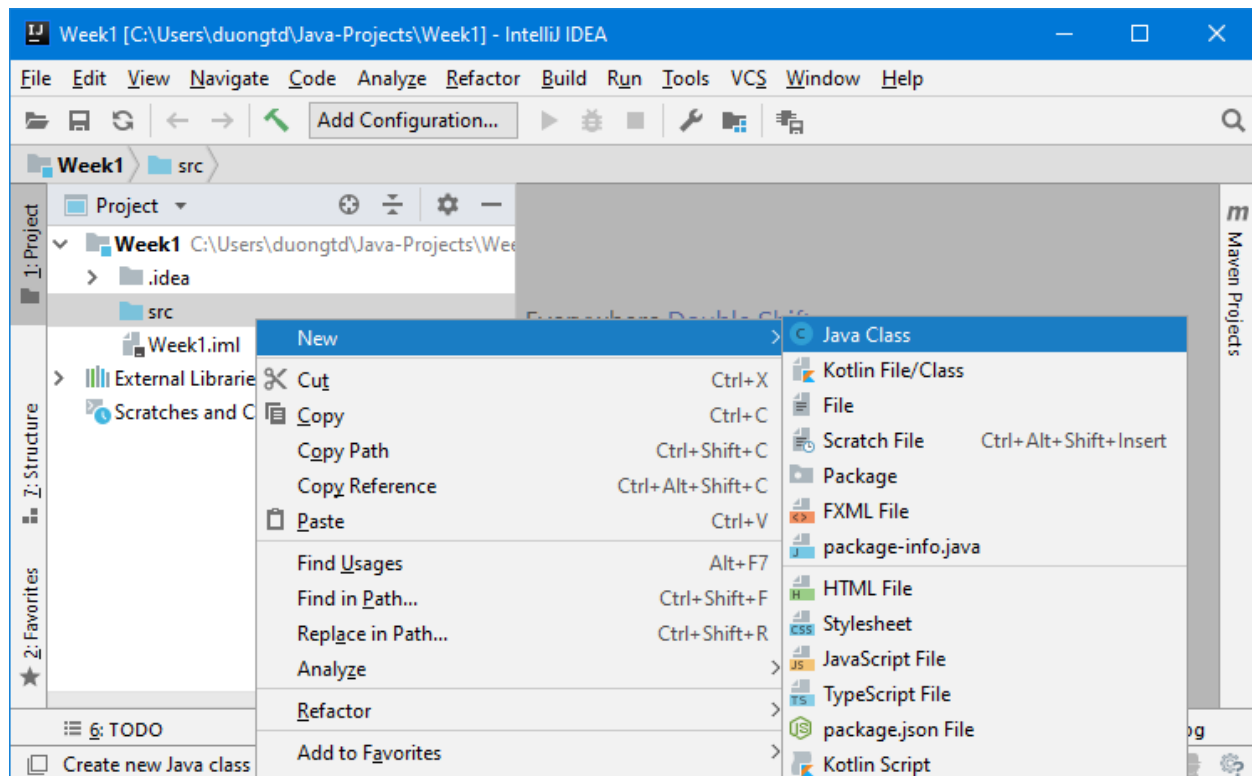


Như vậy là đã config xong, lưu ý việc này chỉ tiến hành 1 lần, các lần sau không cần làm lại.

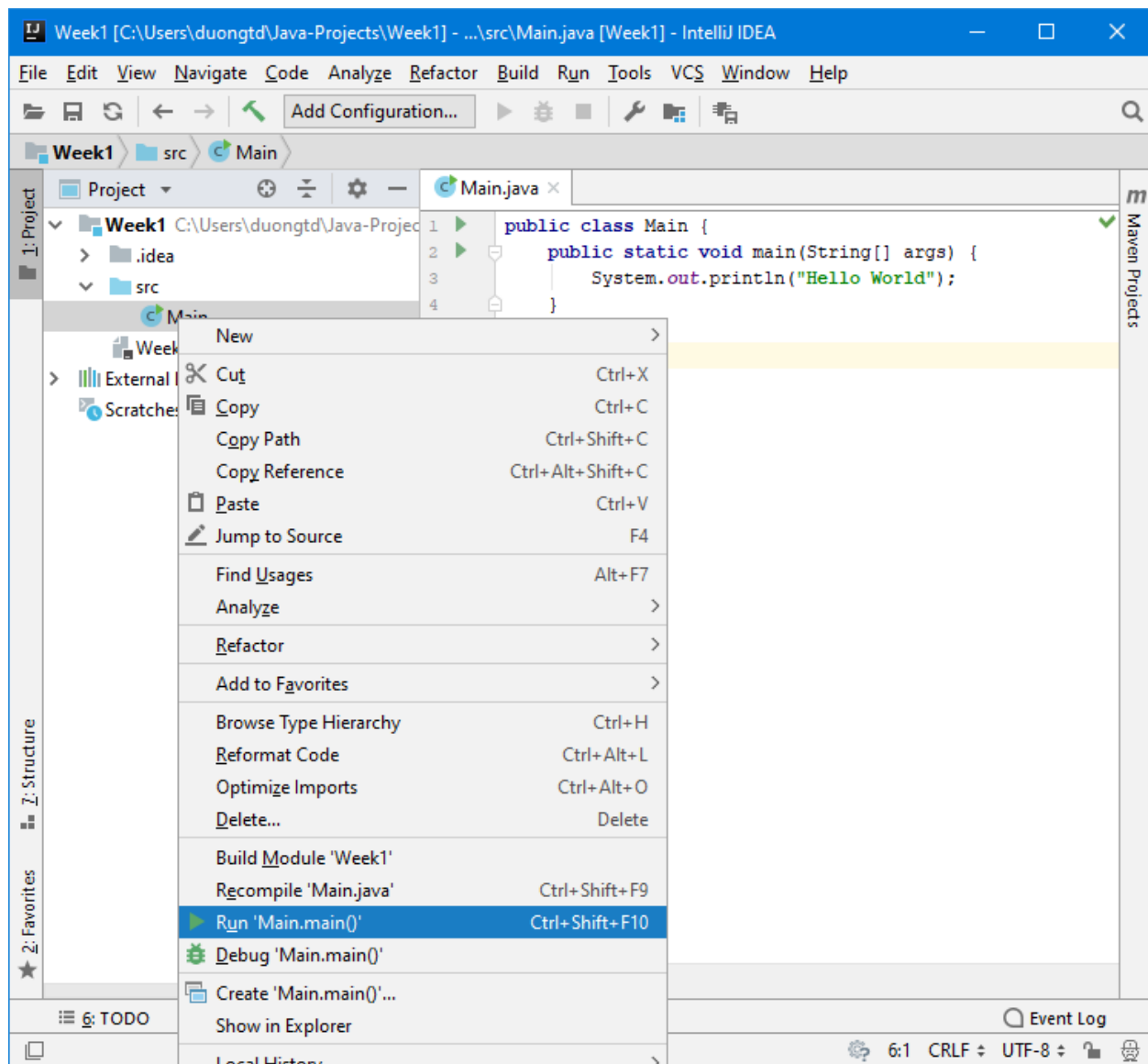
Tiếp theo, để tạo mới project, click vào *Create New Project* ở cửa sổ đầu tiên, tiếp đó 2 lần để nguyên mặc định và click *Next*, *Next* (2 lần). Ở cửa sổ cuối cùng, yêu cầu điền tên project và nơi lưu mã nguồn tương ứng. Sau khi hoàn thành, click *Finish* để xác nhận tạo mới project.



Tạo mới lớp Student bằng cách: right click *src* -> *New* -> *Java Class*; sau đó đặt tên cho class là Student và click *OK*. Tự hoàn thiện các yêu cầu còn lại.

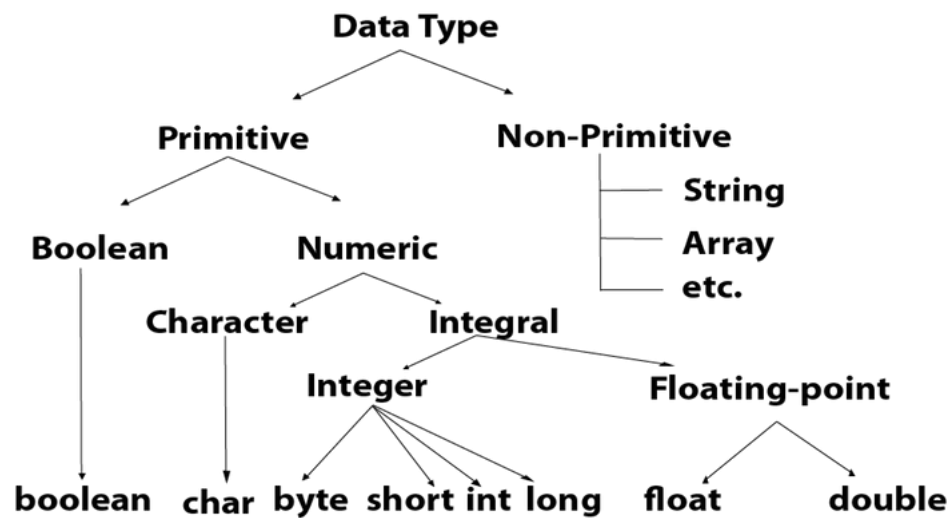


Để chạy chương trình, right click class chứa phương thức *main* cần chạy, chọn *Run* (hoặc *Debug*)



Tuần 3. Kiểu dữ liệu nguyên thủy, equals

Các kiểu dữ liệu trong Java được thể hiện như hình dưới:



Các kiểu dữ liệu trong Java – nguồn javatpoint.com

Sử dụng code mẫu tại <https://github.com/oopuet/Exercises/tree/master/Week3>, hãy hoàn thiện các yêu cầu sau:

Câu 1. Viết các phương thức sau:

- Viết phương thức tìm ước số chung lớn nhất của 2 số nguyên a và b. Đặt tên là **gcd(int a, int b)**.
- Viết phương thức tính Fibonacci của một số nguyên n, đặt tên là **fibonacci(int n)**, công thức như sau:

$$F_n := F(n) := \begin{cases} 0, & \text{khi } n = 0; \\ 1, & \text{khi } n = 1; \\ F(n-1) + F(n-2) & \text{khi } n > 1. \end{cases}$$

- Viết phương thức tìm các số nguyên tố từ 1 -> N (dùng sàng Eratosthenes [wiki](https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes)), đặt tên **sieveEratosthenes(int n)**. Lưu ý, phương thức có kiểu trả về là **void**, tất cả các số nguyên tố tìm được in ra trên cùng một dòng, ngăn cách nhau bởi 1 dấu cách (space).

Câu 2. Tạo lớp phân số **Fraction** có hai thuộc tính là tử (numerator) và mẫu (denominator)

- Khai báo hai thuộc tính **numerator** và **denominator** có kiểu là **int**
- Viết các phương thức getter, setter cho các thuộc tính
- Viết phương thức khởi tạo có tham số cho lớp Fraction có sử dụng từ khóa **this**
- Viết phương thức rút gọn (**reduce**) phân số. Gợi ý sử dụng phương thức tìm ước số chung lớn nhất.

e. Xây dựng các phương thức cộng (**add**), trừ (**subtract**), nhân (**multiply**), chia (**divide**) phân số. Lưu ý các phương thức này trả về 1 phân số mới

f. Viết phương thức “**public boolean equals(Object obj)**” so sánh hai phân số. Nếu **obj** không phải là kiểu **Fraction** thì trả về **false**. Ngược lại trả về **true** khi 2 phân số bằng nhau, còn lại là **false**. Gợi ý sử dụng `instanceof` để kiểm tra **obj** có phải kiểu **Fraction**, sau đó ép kiểu trước khi so sánh.

```
if (obj instanceof Fraction) {  
    Fraction other = (Fraction) obj;  
    // compare this vs other here  
}
```

Tuần 4. Static, mảng, Junit

Viết các phương thức static sau, sử dụng JUnit viết từng phương thức ít nhất 5 bộ test để kiểm tra tính đúng đắn (sử dụng phương thức `assertEquals`). Sử dụng code mẫu tại: <https://github.com/oopuet/Exercises/tree/master/Week4>

- Tìm giá trị lớn nhất của hai số nguyên, giá trị trả về của phương thức là số lớn nhất (phương thức `max2Int`)
- Tìm giá trị nhỏ nhất của của một mảng số nguyên (kích thước mảng ≤ 100 phần tử) (phương thức `minArray`)
- Viết chương trình tính chỉ số BMI theo công thức sau: (phương thức `calculateBMI`)

$$\text{BMI} = \text{Cân nặng (kg)} / (\text{Chiều cao(m)}^2)$$

(Làm tròn kết quả tính được đến chữ số thập phân thứ nhất).

In ra kết quả đánh giá chỉ số BMI dựa theo công thức trên:

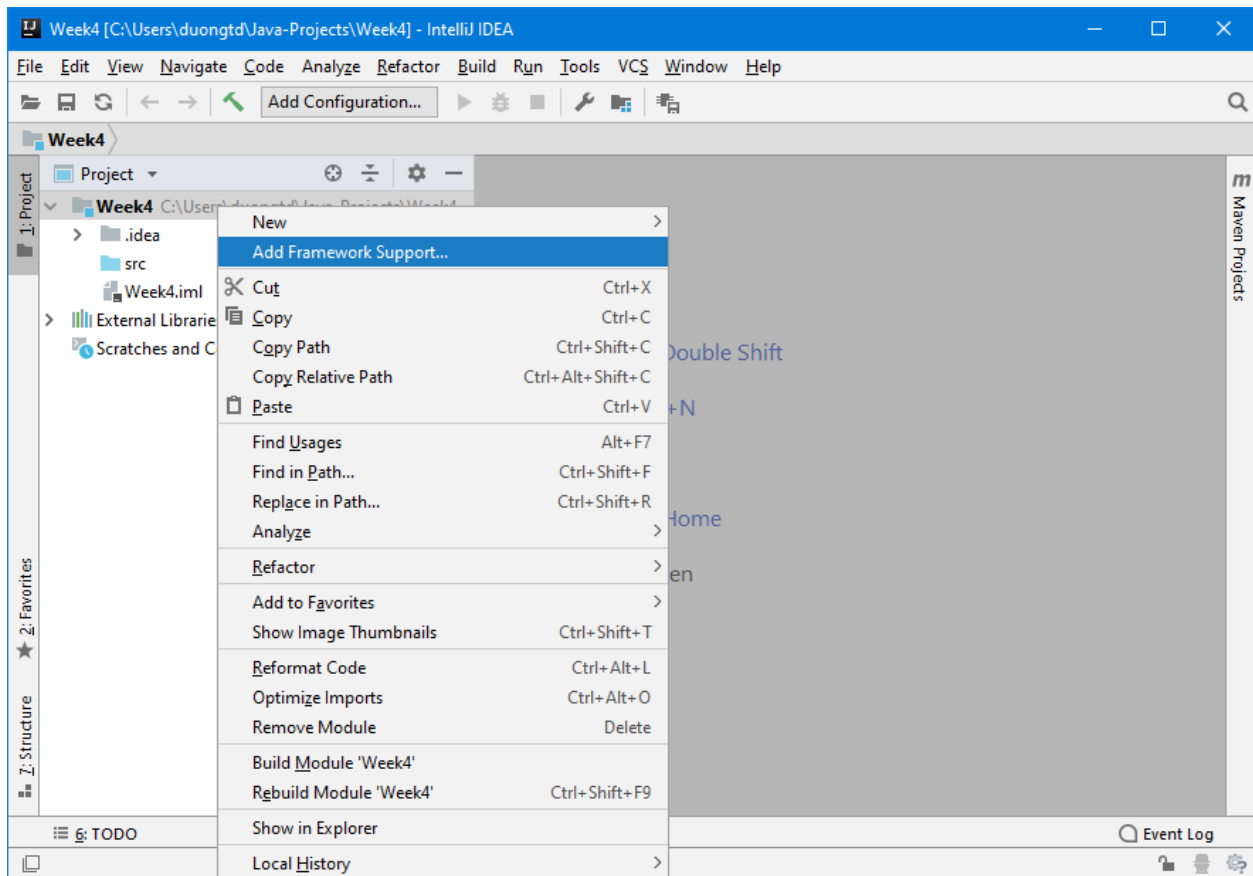
- Nếu BMI dưới 18.5 thì trả về “Thiếu cân”
- Nếu BMI từ 18.5 đến 22.9 thì trả về “Bình thường”
- Nếu BMI từ 23 đến 24.9 thì trả về “Thừa cân”
- Nếu BMI ≥ 25 thì trả về “Béo phì”

Hướng dẫn:

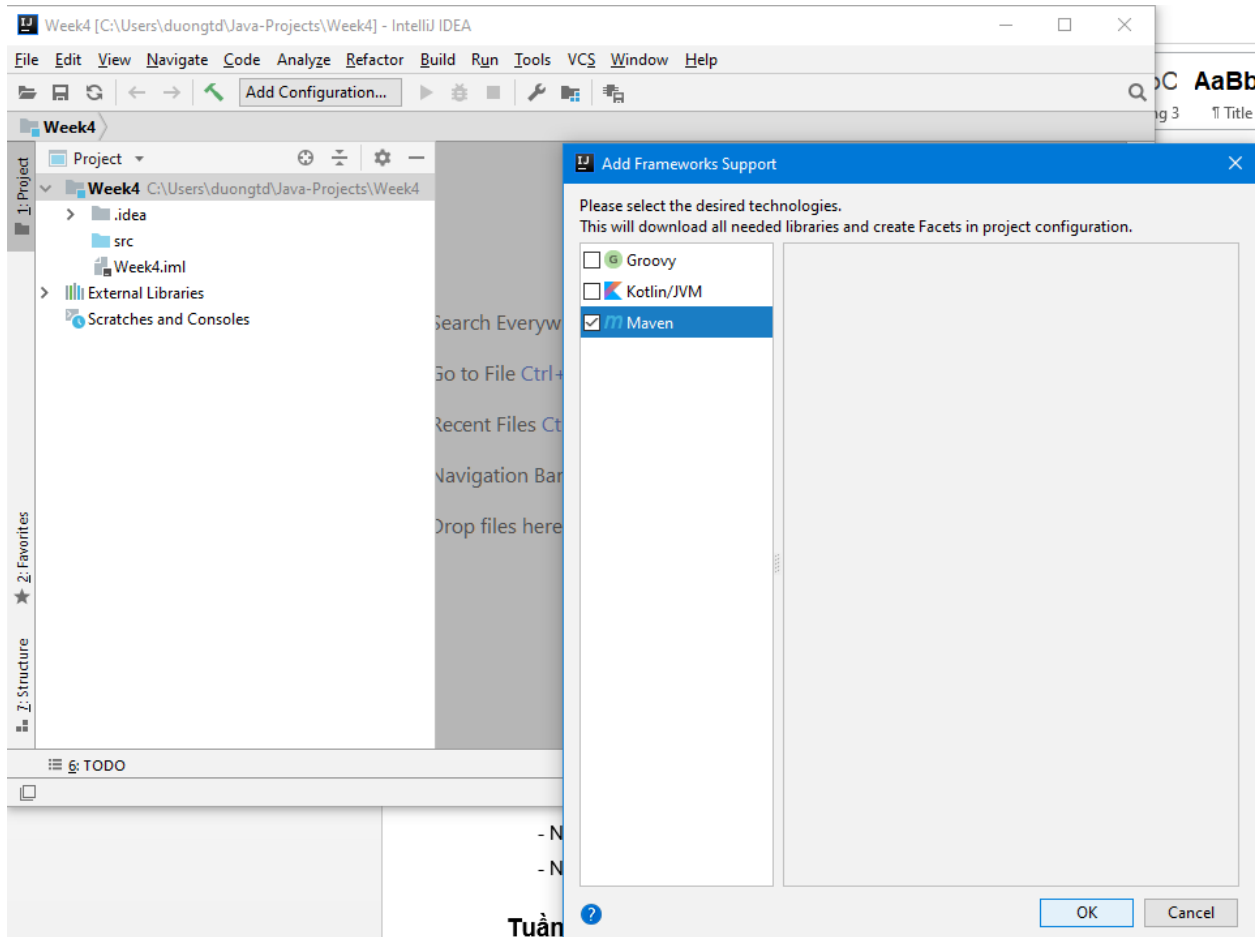
Có nhiều cách khác nhau để thêm thư viện JUnit vào project, để từ đó có thể sử dụng JUnit viết các test case. Hướng dẫn này giới thiệu cách sử dụng Maven để thêm thư viện JUnit vào project.

Đầu tiên giới thiệu qua về Maven, đây là một chương trình quản lý project cho phép developers có thể quản lý về version, các dependencies (các thư viện sử dụng trong dự án), quản lý build, tự động download javadoc & source,... bạn hoàn toàn có thể download thủ công các thư viện (.jar), ví dụ như JUnit, sau đó thêm thủ công vào project. Tuy nhiên, khi số lượng các thư viện dependencies lớn, công việc import thủ công là rất vất vả, chưa kể đến việc version của các thư viện có thể conflict với nhau. Maven có thể giải quyết vấn đề này.

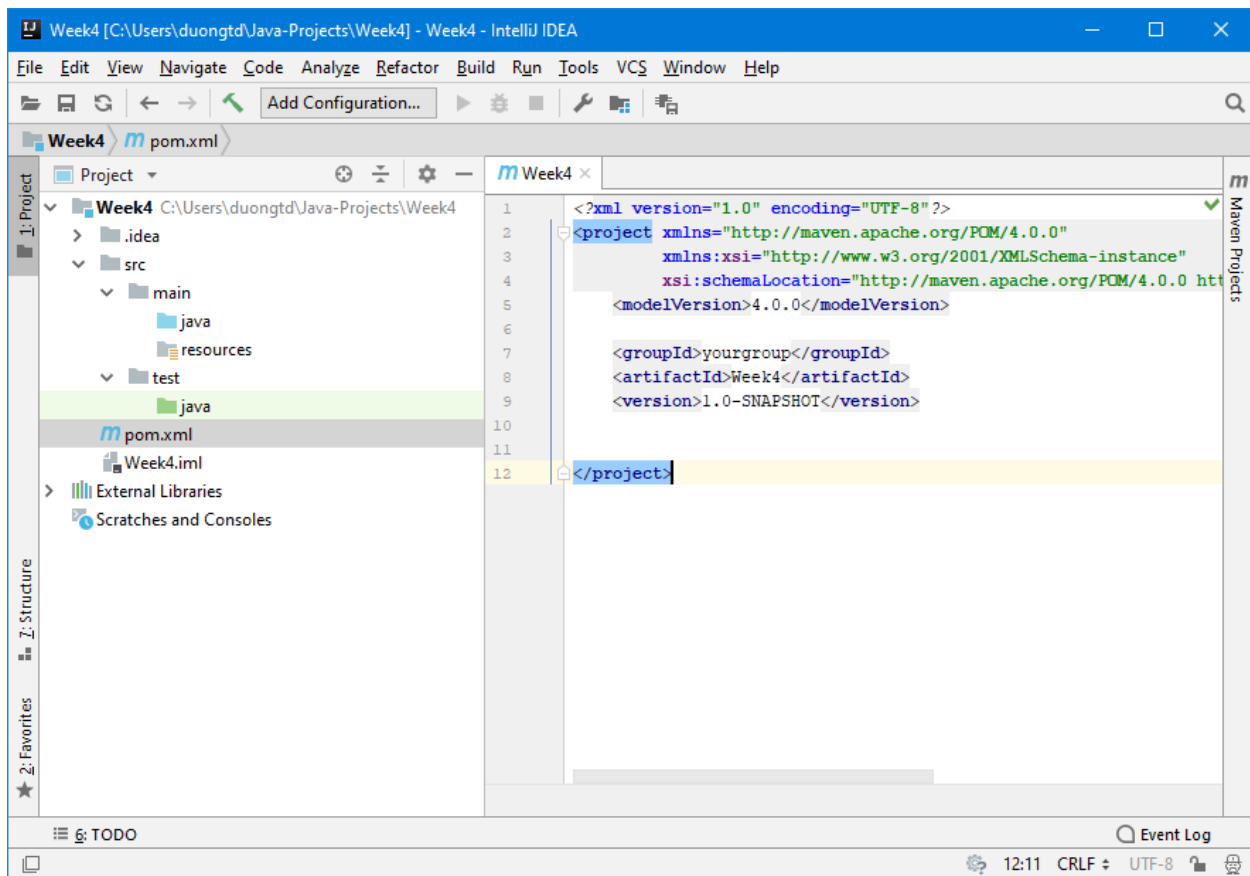
Để sử dụng Maven trong IntelliJ, có 2 cách. Cách 1, trong quá trình tạo mới project (*File -> New -> Project*), bạn chọn đúng kiểu project là Maven trong cửa sổ hiện ra. Cách 2, từ một project Java thông thường như các tuần trước, IntelliJ cho phép bạn convert sang Maven project bằng cách right click project -> *Add Framework Support*



Sau đó, trong cửa sổ hiện ra tích chọn Maven, và click OK để xác nhận.



Sau khi hoàn tất, project đã được chuyển sang Maven project, trong đó có chứa file *pom.xml* – đây là file khai báo các cấu hình (config) cho project của bạn, trong đó có chứa các thư viện sẽ sử dụng (dependencies)



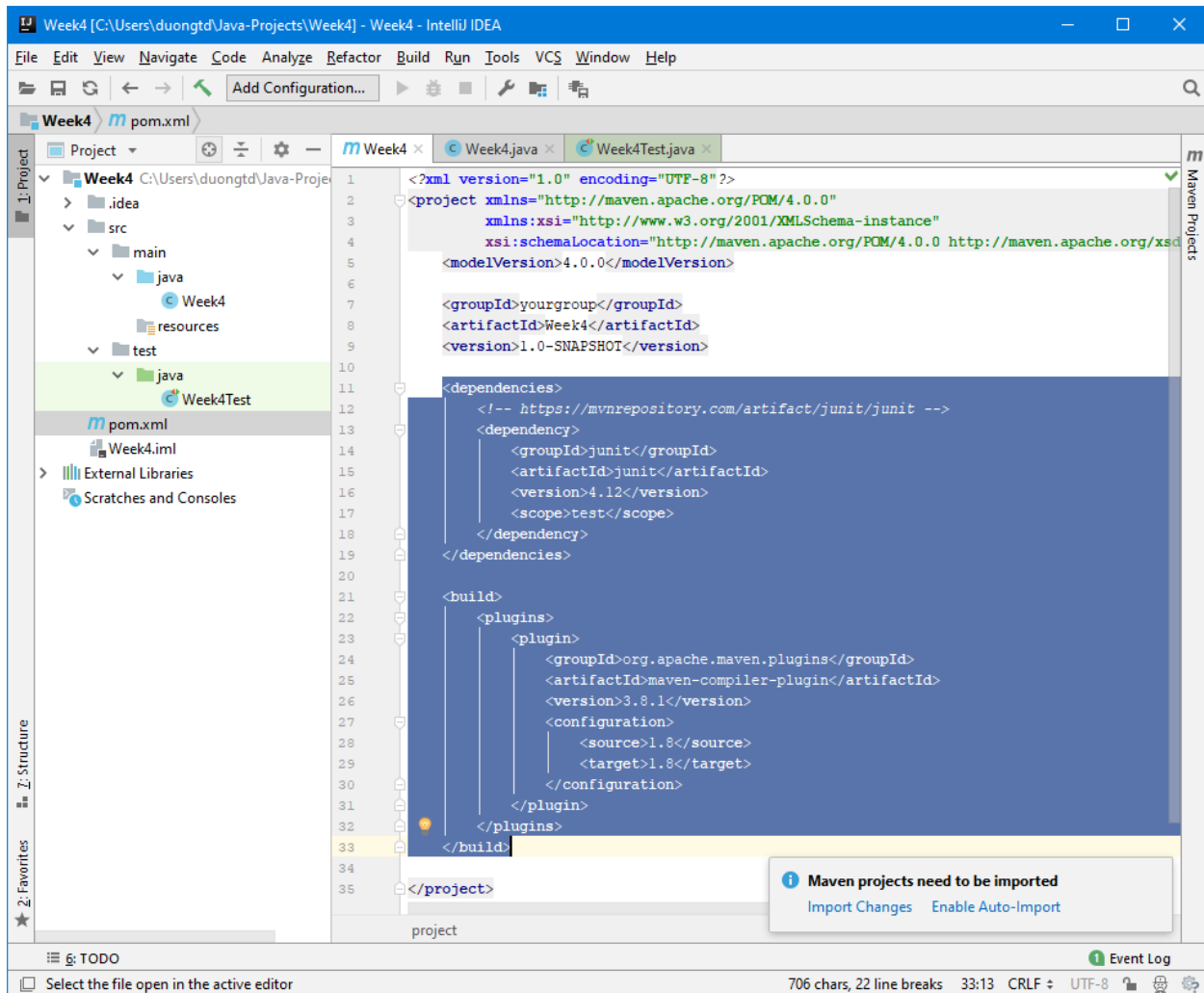
Tiếp theo, để thêm thư viện JUnit 4.2, các bạn chép đoạn mã sau đặt vào trong thẻ `<dependencies>` `</dependencies>` (lưu ý, nếu chưa có thẻ này thì tự tạo thêm thẻ `dependencies` nằm trong thẻ `project`).

```
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

Thêm nữa, bạn cần thêm đoạn mã config Maven Compiler Plugin như dưới (đặt trong thẻ `project`).

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Cuối cùng, click vào *Import Changes* được hiện ra ở cuối sau khi bạn chỉnh sửa file *pom.xml*, thư viện JUnit sẽ được thêm vào project. Hoặc cách khác right click project -> *Maven* -> *Reimport*.

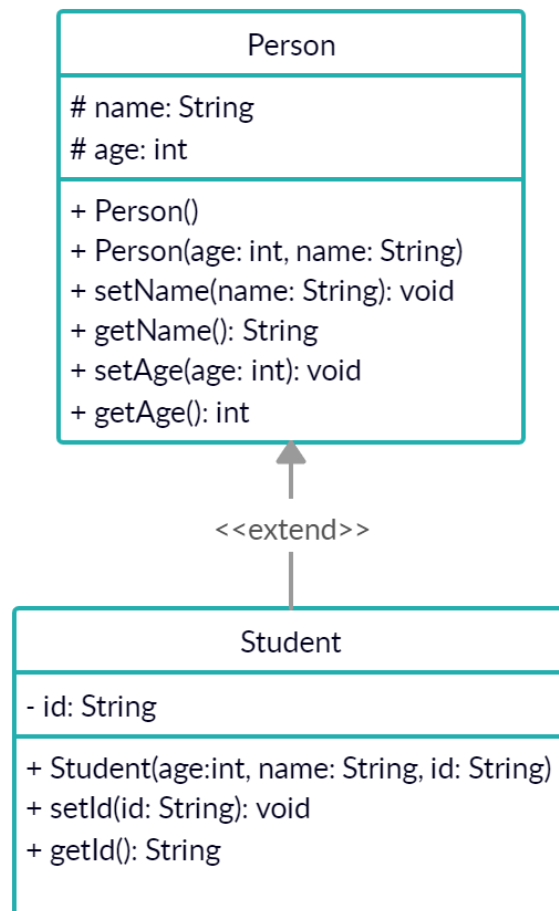


Tới đây, bạn có thể thêm các file test (đặt ở *src* -> *test* -> *java*) và hoàn thiện các yêu cầu còn lại. Chú ý, với tuần này, bạn cần tuân thủ theo code mẫu (link ở trên), trong đó mã nguồn cài đặt gồm 3 hàm static đặt trong class *Week4*; mã nguồn test gồm 5x3=15 hàm đặt trong class *Week4Test*, sử dụng JUnit4, và không thay đổi cấu trúc trong file test này (tên hàm, annotation,...), ngược lại submit sẽ báo sai.

Tuần 5. Kế thừa, final

Biểu đồ lớp (class diagram) dưới đây thể hiện mối quan hệ kế thừa giữa 2 lớp, trong đó lớp Student (sub-class) kế thừa lớp Person (super-class). Biểu đồ được xây dựng theo chuẩn UML2: https://en.wikipedia.org/wiki/Class_diagram. Trong biểu đồ này:

- Lớp Person có:
 - Thuộc tính name, mức truy nhập protected (dấu #), kiểu String
 - Thuộc tính age, mức truy nhập protected (dấu #), kiểu int
 - Phương thức khởi tạo không tham số Person(), mức truy nhập public (dấu +)
 - Phương thức khởi tạo 2 tham số public Person(int age, String name)
 - Các phương thức public getter/setter cho 2 thuộc tính name và age
- Lớp Student kế thừa lớp Person, có:
 - Thuộc tính id, mức truy nhập private, kiểu String
 - Phương thức khởi tạo 3 tham số public Person(int age, String name, String id)
 - Các phương thức public getter/setter cho thuộc tính id



Khi cài đặt code cho biểu đồ lớp bên trên, mã nguồn tương ứng như sau:

```

//file Person.java
public class Person {
    protected int age;
    protected String name;

    public Person() {
    }

    public Person(int age, String name) {
        this.age = age;
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

//file Student.java
public class Student extends Person {
    private String id; //MSSV

    public Student(int age, String name, String id) {
        super(age, name);
        this.id = id;
    }

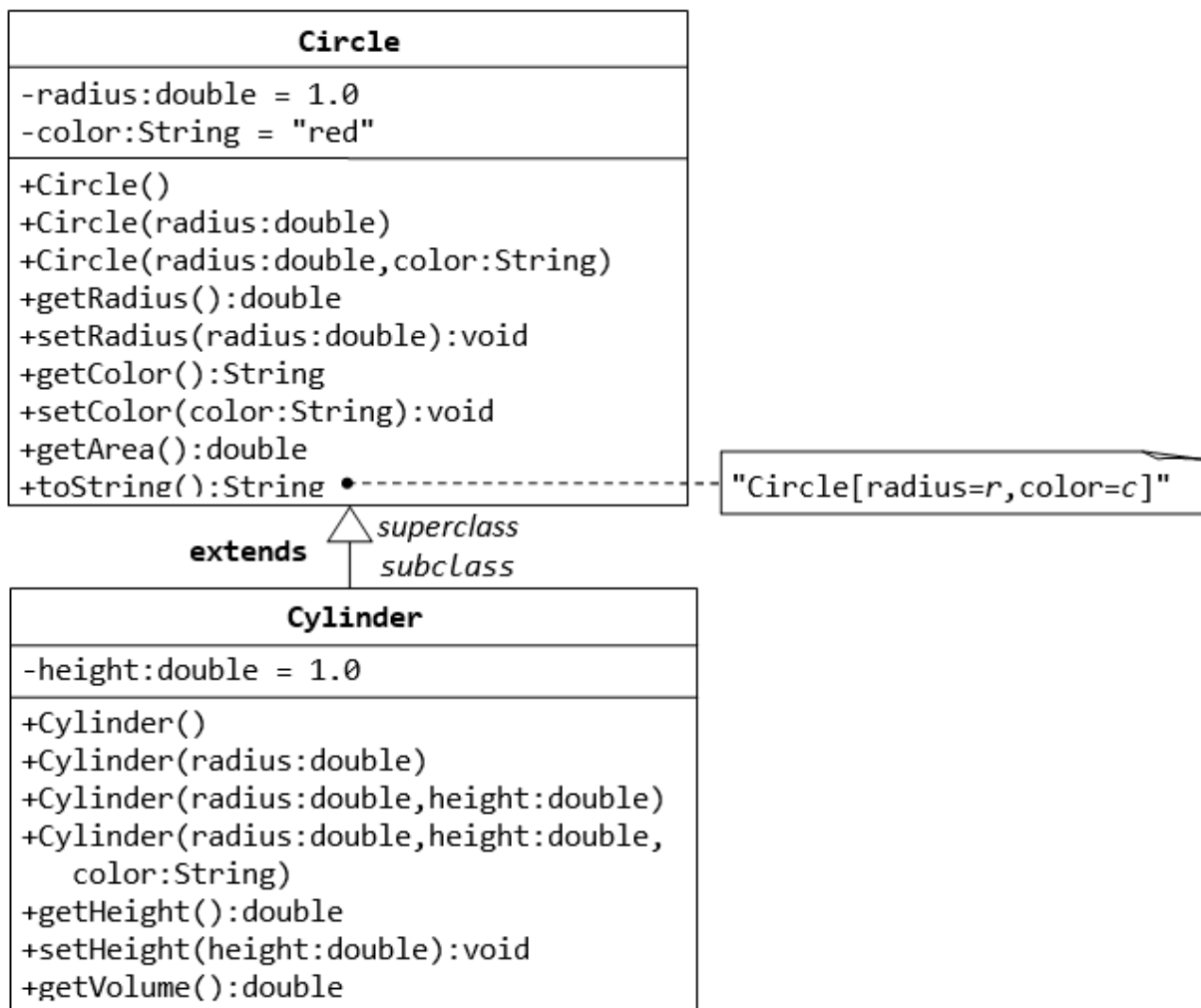
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }
}

```

Yêu cầu: Dựa vào hướng dẫn bên trên, hãy hoàn thành 2 yêu cầu dưới đây.

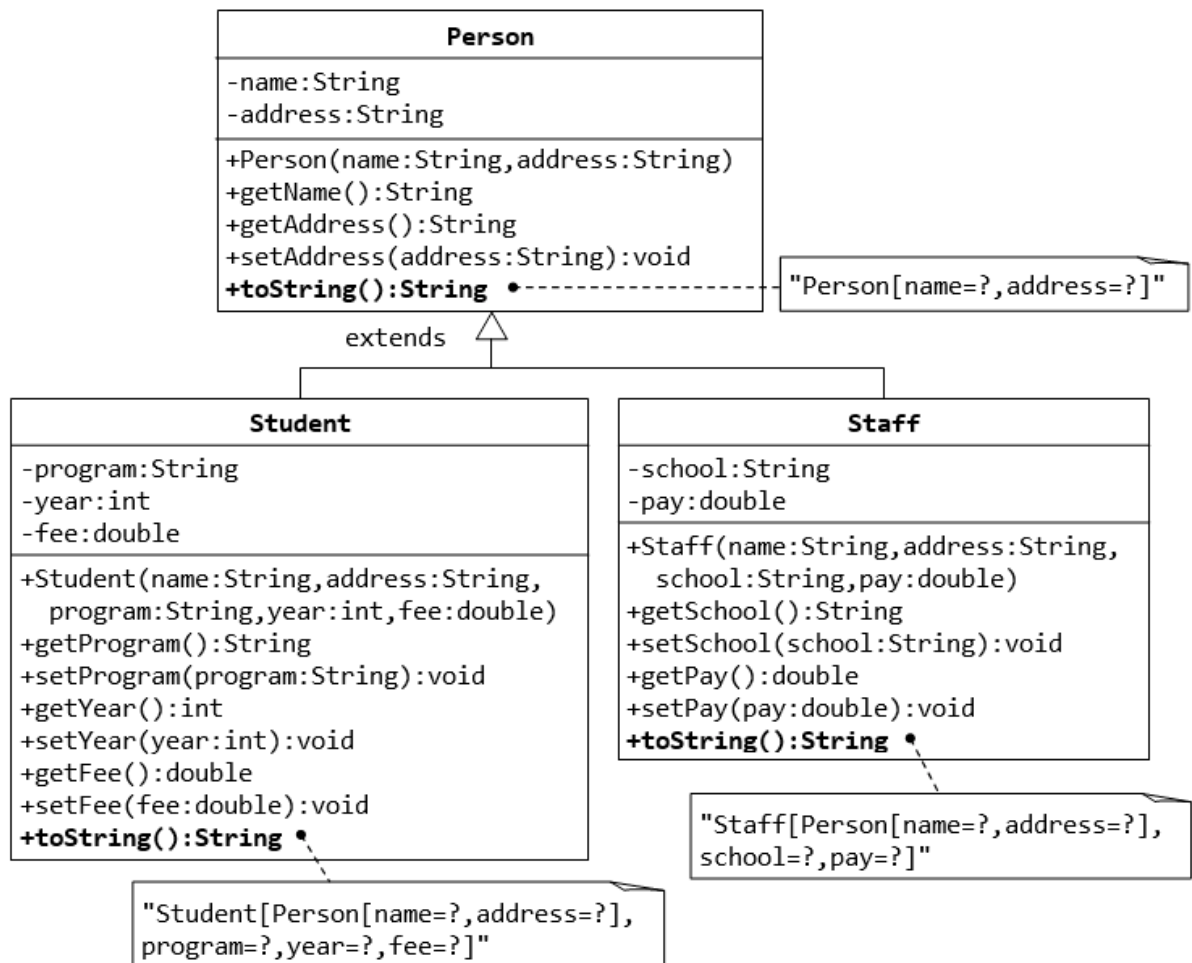
Câu 1. Cài đặt 2 lớp **Circle** và **Cylinder** tuân theo biểu đồ lớp dưới đây:



Sau khi cài đặt xong, thực hiện tiếp các yêu cầu dưới đây:

- Sử dụng **final** khai báo hằng số $\pi = 3.14$, có tên là **PI**, kiểu dữ liệu **double**, mức truy cập **protected** để các phương của lớp **Cylinder** có thể sử dụng được. Các phương thức tính diện tích, thể tích sử dụng giá trị của hằng số **PI** này để tính toán.
- Override phương thức **toString()** cho lớp **Cylinder**, trả về giá trị theo định dạng `Cylinder[Circle[radius=r,color=c],height=h]`
Gợi ý tái sử dụng phương thức **toString()** của lớp **Circle** thông qua từ khóa **super** để xây dựng phương thức **toString()** này
- Override phương thức **getArea()** cho lớp **Cylinder**
Gợi ý sử dụng phương thức **getArea()** của lớp **Circle** để tính diện tích cho hai mặt đáy

Câu 2. Cài đặt các lớp **Student**, **Staff**, **Person** tuân theo biểu đồ lớp dưới đây:



Tuần 6. (1) Đa hình, instanceof, abstract

Đa hình trong Java được hiểu là một đối tượng có thể có nhiều hình thức, kiểu khác nhau. Đầu tiên để hiểu được đa hình, khái niệm up-casting sẽ được làm rõ. Up-casting là khả năng nhìn nhận đối tượng thuộc con như là một đối tượng thuộc lớp cha. Vẫn sử dụng lại ví dụ ở tuần 5 về 2 lớp Person và Student, trong đó Student kế thừa Person, đoạn code dưới đây minh họa rõ về up-casting. *Lưu ý, những đoạn code không liên quan được bỏ qua, không thể hiện tất cả ở đây.*

```
public class Student extends Person {  
  
    ... // bỏ qua code cũ  
  
    public static void main(String[] args) {  
        // up-casting, biến person của lớp Person (lớp cha) tham  
        // chiếu tới đối tượng của lớp Student (lớp con)  
        Person person = new Student(20, "An", "18020000");  
    }  
}
```

Bây giờ, thêm vào lớp Person phương thức printType() để in ra đây là đối tượng thuộc lớp Person; đồng thời, lớp Student override phương thức này, in ra đây là đối tượng thuộc lớp Student. Code sau khi đã chỉnh sửa có dạng như dưới đây.

```
public class Person {  
  
    ... // bỏ qua code cũ  
  
    public void printType() {  
        System.out.println("This is a person");  
    }  
}  
  
public class Student extends Person {  
  
    ... // bỏ qua code cũ  
  
    public void printType() {  
        System.out.println("This is a student");  
    }  
  
    public static void main(String[] args) {  
        // up-casting, biến person của lớp Person (lớp cha) tham  
        // chiếu tới đối tượng của lớp Student (lớp con)  
        Person person = new Student(20, "An", "18020000");  
        person.printType(); // This is a student  
    }  
}
```

Khi thực thi hàm main bên trên, chương trình sẽ in ra "This is a student". Điều đó có nghĩa là dù ta gọi phương thức printType() bởi biến tham chiếu person thuộc lớp Person (lớp cha),

phương thức ghi đè `printType()` thuộc lớp `Student` (lớp con) vẫn được gọi tới. Đó chính là tính đa hình trong Java.

Tiếp theo, trong Java cung cấp toán tử `instanceof` để kiểm tra xem đối tượng có là instance của kiểu cụ thể: lớp hoặc lớp con hoặc interface hay không. Nó trả về `true` hoặc `false`. Ví dụ dưới đây minh họa cụ thể về toán tử này.

```
Person person = new Student(20, "An", "18020000");
boolean b1 = person instanceof Person; // true
boolean b2 = person instanceof Student; // true

Person person2 = new Person();
boolean b3 = person2 instanceof Person; // true
boolean b4 = person2 instanceof Student; // false
```

Tính trừu tượng (Abstraction) trong Java hướng đến khả năng tạo một đối tượng trừu tượng trong lập trình hướng đối tượng. Một lớp trừu tượng (abstract class) là một lớp mà không thể được khởi tạo. Tất cả các chức năng khác của lớp, thuộc tính, phương thức, và hàm khởi tạo đều giống với lớp thông thường. Chỉ khác là ta không thể tạo một đối tượng với một lớp trừu tượng hóa. Trong Java, ta sử dụng từ khóa `abstract` để khai báo một lớp abstract. Từ khóa này xuất hiện trước từ khóa `class` trong khai báo lớp.

Trong ví dụ dưới đây, lớp `Bank` là 1 lớp trừu tượng (không thể khởi tạo đối tượng `Bank`), lớp này có khai báo 1 phương thức trừu tượng là `getRateOfInterest()` – trả về lãi suất của bank tương ứng. Vì chưa rõ là loại ngân hàng nào, lãi suất này với mỗi ngân hàng khác nhau là khác nhau, nên phương thức này cần khai báo kiểu trừu tượng mà không cài đặt ngay. 2 lớp `VCB` (Vietcombank) và `BIDV` kế thừa lớp `Bank`. Vì vậy chúng bắt buộc phải cài đặt phương thức `getRateOfInterest()`. Đây chính là ý nghĩa của trừu tượng trong Java.

```
public abstract class Bank {
    abstract double getRateOfInterest();
}

public class VCB extends Bank {
    @Override
    double getRateOfInterest() {
        return 7.1;
    }
}

public class BIDV extends Bank {
    @Override
    double getRateOfInterest() {
        return 7.2;
    }
}

public class TestBank {
    public static void main(String[] args) {
        Bank bank = new BIDV();
        double interest = bank.getRateOfInterest();
    }
}
```

```

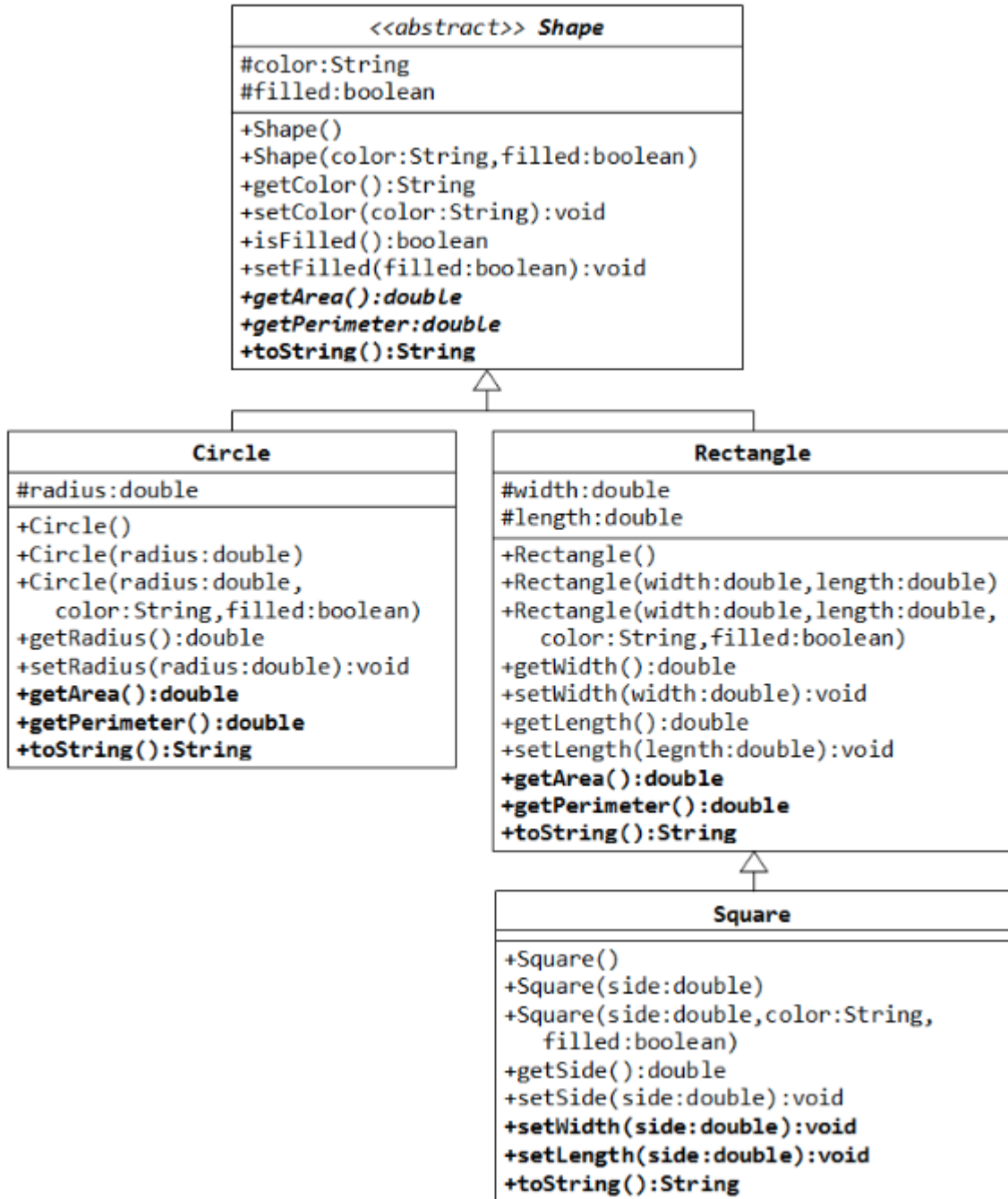
        System.out.println("Ti le lai suat la: " + interest +
        " %");
    }
}

```

Lỗi sẽ xảy ra nếu ta cố tình khởi tạo trực tiếp đối tượng từ lớp Bank (thay vì lớp con của nó).

```
Bank b2 = new Bank(); // error
```

Yêu cầu: Dựa vào hướng dẫn bên trên, hãy cài đặt các lớp theo biểu đồ dưới đây.



Một vài điều lưu ý sau đây:

1. Các thuộc tính với dấu # thể hiện mức truy nhập *protected*
 2. Shape là abstract class, có 2 phương thức abstract là *getArea()* và *getPerimeter()* lần lượt tương ứng trả về diện tích và chu vi của hình đó.
 3. Các phương thức *toString()* của 4 lớp quy định như sau:
 - Shape: Shape[color=?,filled=?]
 - Circle: Circle[radius=?,color=?,filled=?]
 - Rectangle: Rectangle[width=?,length=?,color=?,filled=?]
 - Square[side=?,color=?,filled=?]
- ❖ Xem xét đoạn code dưới đây, trước khi test chạy thử, hãy tìm và dự đoán xem dòng nào sẽ báo lỗi, và kịch bản in ra sẽ như thế nào. *Lưu ý, có thể 1 số câu lệnh có lỗi syntax, dẫn đến compile error:*

```
Shape s1 = new Circle(5.5, "RED", false);
System.out.println(s1);
System.out.println(s1.getArea());
System.out.println(s1.getPerimeter());
System.out.println(s1.getColor());
System.out.println(s1.isFilled());
System.out.println(s1.getRadius());

Circle c1 = (Circle)s1;
System.out.println(c1);
System.out.println(c1.getArea());
System.out.println(c1.getPerimeter());
System.out.println(c1.getColor());
System.out.println(c1.isFilled());
System.out.println(c1.getRadius());

Shape s2 = new Shape();

Shape s3 = new Rectangle(1.0, 2.0, "RED", false);
System.out.println(s3);
System.out.println(s3.getArea());
System.out.println(s3.getPerimeter());
System.out.println(s3.getColor());
System.out.println(s3.getLength());

Rectangle r1 = (Rectangle)s3;
System.out.println(r1);
System.out.println(r1.getArea());
System.out.println(r1.getColor());
```

```
System.out.println(r1.getLength());
```

```
Shape s4 = new Square(6.6);  
System.out.println(s4);  
System.out.println(s4.getArea());  
System.out.println(s4.getColor());  
System.out.println(s4.getSide());
```

```
Rectangle r2 = (Rectangle)s4;  
System.out.println(r2);  
System.out.println(r2.getArea());  
System.out.println(r2.getColor());  
System.out.println(r2.getSide());  
System.out.println(r2.getLength());
```

```
Square sq1 = (Square)r2;  
System.out.println(sq1);  
System.out.println(sq1.getArea());  
System.out.println(sq1.getColor());  
System.out.println(sq1.getSide());  
System.out.println(sq1.getLength());
```

Tuần 7. (2) Đa hình, interface

Tiếp tục từ bài buổi trước, hãy:

- Định nghĩa lớp **Diagram** đại diện cho sơ đồ đang được vẽ
- Định nghĩa lớp **Layer** chứa danh sách các đối tượng **Shape**. Lớp **Diagram** có một hoặc nhiều đối tượng thuộc lớp **Layer**.
- Bổ sung phương thức cho lớp **Layer** để xóa tất cả các đối tượng thuộc lớp **Triangle** trong lớp
- Bổ sung phương thức cho lớp **Diagram** để xóa tất cả các đối tượng thuộc lớp **Circle** trong **Diagram**
- Thêm thuộc tính visible cho Layer. Khi thuộc tính này là false, các hình thuộc đối tượng Layer đấy sẽ không được vẽ trên Diagram
- Viết phương thức xóa các hình trùng nhau trong một Layer (ví dụ: với Circle, 2 hình trùng nhau sẽ tọa độ tâm và độ lớn bán kính như nhau)
- Viết phương thức để chuyển từng loại hình vẽ vào từng đối tượng Layer. Sau khi chạy phương thức này, tất cả các Square của Diagram sẽ được chuyển vào 1 Layer, các Circle được chuyển vào 1 Layer khác,... Nếu số loại hình nhiều hơn số Layer hiện có thì tạo thêm.
- Từ các lớp đã cài đặt sẵn, thử khởi tạo các đối tượng, render đồ họa cho chúng, gợi ý trong Java, có thể sử dụng Swing or JavaFX cho đồ họa.

Tuần 8. Ngoại lệ (try-catch, throw)

Trong Java, ngoại lệ - exception là một sự kiện mà phá vỡ luồng chuẩn trong quá trình thực hiện của chương trình. Nhiệm vụ của ta – người phát triển chương trình là xử lý ngoại lệ, thay vì chương trình “chết” thì chủ động xử lý để nó có thể tiếp tục bình thường. Để dễ tiếp cận, hãy xem xét ví dụ đơn giản dưới đây.

```
public class HelloException {
    public static void main(String[] args) {
        // Phép chia này không có vấn đề
        double value = 4 / 2;
        System.out.println("Value = " + value);

        // Phép chia này có vấn đề, vì chia cho 0
        // Lỗi đã xảy ra tại đây.
        value = 2 / 0;

        // 2 dòng code dưới đây sẽ không được thực hiện
        System.out.println("New value = " + value);
        System.out.println("Exit!");
    }
}
```

Khi thực thi đoạn code này, kết quả thu được như sau:

```
Value = 2
Exception in thread "main" java.lang.ArithmeticException: / by zero
at exception>HelloException.main(HelloException.java:12)
```

Trong ví dụ này, lỗi xảy ra khi thực hiện phép chia cho 0. Phép chia cho 0 làm chương trình “ném” ra ngoại lệ, “chết” ngay tại dòng này, dẫn tới 2 câu lệnh cuối cùng không được thực hiện. Tiến hành sửa đổi đoạn code thành như dưới đây, và chạy lại chương trình, ta sẽ thấy chương trình không báo “lỗi”, câu lệnh cuối đã được thực hiện.

```
public class HelloException {
    public static void main(String[] args) {
        // Phép chia này không có vấn đề
        double value = 4 / 2;
        System.out.println("Value = " + value);

        try {
            value = 2 / 0;
            System.out.println("New value = " + value);
        } catch (ArithmeticException e) {
            // Các dòng code trong khối catch được thực thi
            System.out.println("An exception");
            System.out.println("Error: " + e.getMessage());
        }

        // Dòng code dưới đây chắc chắn sẽ được thực hiện
        System.out.println("Exit!");
    }
}
```

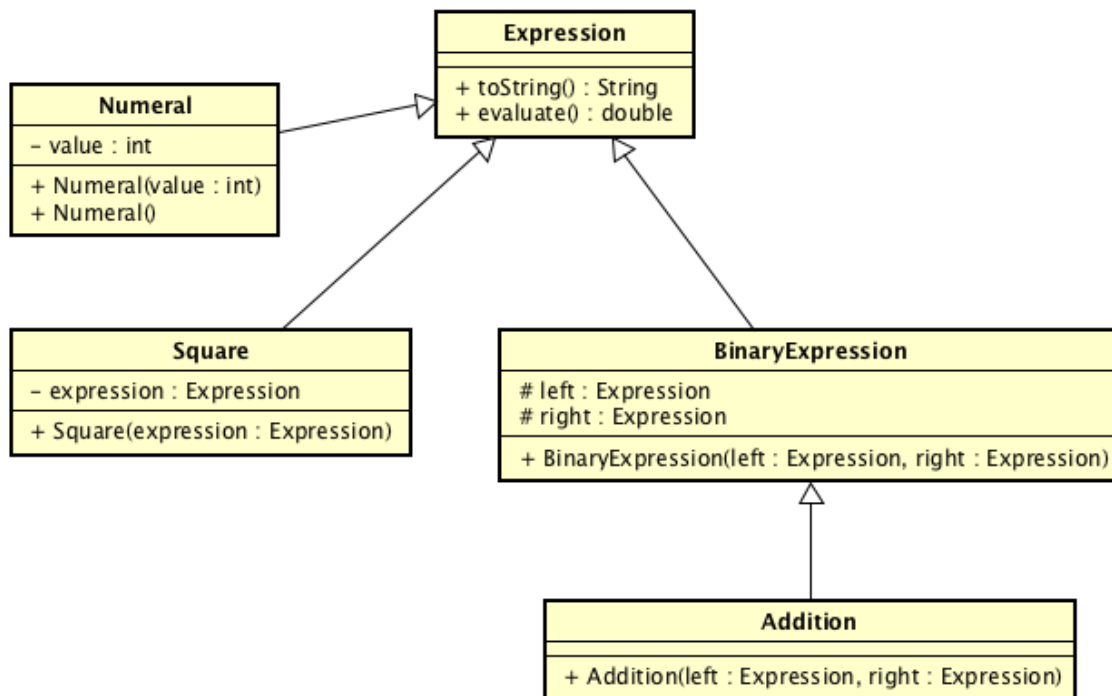
Ở đoạn code sửa đổi phía dưới, ta đã chủ động đặt đoạn code có thể xảy ra lỗi (thực hiện phép chia) vào trong khối try-catch. Vì vậy khi ngoại lệ được “ném” ra, đoạn code trong khối catch sẽ được thực thi, sau đó tiếp tục với các lệnh còn lại.

Ví dụ xử lý ngoại lệ bên trên có thể sửa đổi thành dạng “chuẩn hơn” như dưới đây. Thông thường, hàm chứa đoạn code có thể xảy ra ngoại lệ sẽ “ném” ra ngoại lệ. Và hàm gọi tới chúng sẽ xử lý chúng bằng cách try-catch hoặc tiếp tục “ném ra ngoại lệ” để hàm “gọi tới nó” xử lý.

```
public class HelloException {  
  
    // phương thức này "ném" ra ngoại lệ, phương thức gọi tới cần xử lý  
    public static double divide(double a, double b) throws  
    ArithmeticException {  
        return a/b;  
    }  
  
    public static void main(String[] args) {  
        try {  
            value = divide(2 , 0);  
            System.out.println("New value = " + value);  
        } catch (ArithmeticException e) {  
            System.out.println("An exception");  
            System.out.println("Error: " + e.getMessage());  
        }  
        System.out.println("Exit!");  
    }  
}
```


Yêu cầu: Dựa trên hướng dẫn phía trên, hãy hoàn thành các yêu cầu dưới đây

Câu 1. Cài đặt các lớp như trong hình dưới đây.



- Đặt modifier abstract cho các lớp và phương thức thích hợp
- Cài đặt các phương thức **evaluate()** và **toString()** thích hợp cho các lớp. Dưới đây là ví dụ kết quả phương thức **toString()** cho biểu thức $(10^2 - 3 + 4 \cdot 3)^2$.

$((((10) ^ 2 + -3) + (4 * 3))) ^ 2$

- Viết lớp **ExpressionTest** để thử, trong đó tạo biểu thức $(10^2 - 3 + 4 \cdot 3)^2$ và tính kết quả, in ra màn hình. Chú ý: không cần nhập dữ liệu từ tệp ngoài hoặc xử lý chuỗi ký tự chứa biểu thức.
- Bổ sung các lớp **Subtraction** (trừ), **Multiplication** (nhân), và **Division** (chia). Sau đó, sửa lớp **ExpressionTest** để chạy thử.
- Xử lý ném ngoại lệ **java.lang.ArithmeticException** với nội dung **"Lỗi chia cho 0"** khi thực hiện phép chia cho 0 ở phương thức **evaluate()** của lớp **Division**

Câu 2. Tạo các phương thức gây ra và xử lý 5 trường hợp ngoại lệ như mô tả ở bảng dưới. *Phương thức ném ngoại lệ* là phương thức sẽ gây ra ngoại lệ; *Phương thức bắt ngoại lệ* là các phương thức gọi đến và xử lý ngoại lệ ném ra từ *Phương thức ném ngoại lệ*. *Phương thức bắt ngoại lệ* sau khi xử lý ngoại lệ xong thì trả về kết quả lỗi tương ứng (cột **Trả về lỗi**, kiểu dữ liệu **String**), trong trường hợp không có ngoại lệ xảy ra thì trả về "Không có lỗi".

Tất cả phương thức viết trong 1 lớp có tên là **Week8_Task2**.

Ngoại lệ	Phương thức ném ngoại lệ	Phương thức bắt ngoại lệ	Lỗi trả về	Hướng dẫn
java.lang.NullPointerException	NPEX	NPEXTest	“Lỗi Null Pointer”	Cố tình truy cập một đối tượng có giá trị <i>null</i>
java.lang.ArrayIndexOutOfBoundsException	AIOoBEx	AIOoBExTest	“Lỗi Array Index Out of Bounds”	Cố tình truy cập vào vị trí chỉ mục không tồn tại trong một mảng
java.lang.ArithmeticException	ArithEx	ArithExTest	“Lỗi Arithmetic”	Thực hiện phép chia cho 0
java.io.FileNotFoundException	FNFEEx	FNFEExTest	“Lỗi File Not Found”	Sử dụng lớp <i>FileReader</i> cố tình đọc nội dung một tệp không tồn tại
java.io.IOException	IOEx	IOExTest	“Lỗi IO”	-

Tuần 9. I/O

Java cung cấp sự hỗ trợ mạnh mẽ cho các thao tác I/O liên quan tới các File (đọc, ghi,...). Ví dụ với đọc 1 file text, Java cung cấp nhiều cách khác nhau. Có thể sử dụng FileReader như dưới đây.

```
BufferedReader in = new BufferedReader(new
FileReader("c:\\filename.txt"));
String str;
while ((str = in.readLine()) != null) {
    System.out.println(str);
}
```

Hoặc có thể sử dụng File.ReadAllLines() như dưới đây. Phương thức này sẽ trả về 1 List các String tương ứng với các dòng của file text.

```
List<String> lines =
Files.readAllLines(Paths.get("C:\\filename.txt"));
```

Yêu cầu: Dựa trên hướng dẫn phía trên, hãy xây dựng lớp tiện ích Utils, sử dụng code mẫu tại: <https://github.com/oopuet/Exercises/tree/master/Week9>

a. Viết phương thức static để đọc một tệp .txt từ ổ cứng:

```
public static String readContentFromFile(String path),
```

trong đó biến path là đường dẫn đến tệp cần đọc, lưu ý cả trường hợp đường dẫn tương đối và tuyệt đối, quản lý file trên hệ điều hành Linux cũng như Windows

b. Viết phương thức static để xuất nội dung một xâu vào 1 tệp trong ổ cứng:

```
public static void writeContentToFile(String path),
```

trong đó biến path là đường dẫn đến tệp cần ghi nội dung. Nếu tệp đã có nội dung thì ta xóa nội dung đó trước khi ghi nội dung mới.

c. Tương tự ý câu b nhưng thay vì xóa nội dung cũ đi, ta bổ sung nội dung mới vào cuối tệp hiện tại

d. Viết phương thức static để tìm kiếm một tệp trong một thư mục:

```
public static File findFileByName(String folderPath, String fileName),
```

trong đó folderPath là tên thư mục, fileName là tên tệp cần tìm kiếm.

Tuần 10. Data structures: List, Array, String

❖ List

List Interface là một loại Interface Collection. Các phần tử của List Interface được sắp xếp có thứ tự và giá trị của các phần tử này có thể trùng nhau. Các phần tử trong List có thể được truy cập, thêm, sửa hoặc xóa thông qua vị trí của nó trong danh sách, và phần tử đầu tiên trong List sẽ có chỉ số là 0. Chi tiết về các phương thức của List có thể xem tại đây: <https://docs.oracle.com/javase/8/docs/api/java/util/List.html>

```
List<String> list = new ArrayList<String>();
list.add("An");
list.add("Nguyen");
list.add("Tung");
System.out.println(list.get(2)); // Tung
list.add(1, "Binh");
System.out.println(list.get(2)); // Nguyen
```

❖ String

Chuỗi (String) trong Java cung cấp nhiều khái niệm đa dạng giúp ta thao tác và xử lý với chuỗi như so sánh, cắt, nối, tìm độ dài, thay thế, tìm chuỗi con,.... Trong Java, về cơ bản chuỗi là một đối tượng mà biểu diễn dãy các giá trị char. Chuỗi (String) trong Java là không thể thay đổi (immutable), ví dụ: nó không thể bị thay đổi nhưng sẽ có một instance được tạo. Tuy nhiên, nếu muốn sử dụng các lớp mà có thể thay đổi, có thể lựa chọn sử dụng các lớp StringBuffer và StringBuilder. Chi tiết về các phương thức của String có thể xem tại đây: <https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>

```
String a = new String("abc");
String b = "abc";
System.out.println(a == b); // false
System.out.println(a.equals(b)); // true
System.out.println(a.length()); // 3
System.out.println(a + b); // abcabc
a = " demo_trim ";

System.out.println("before trim:" + a + "!");
// before trim: demo_trim !

System.out.println("after trim:" + a.trim() + "!");
// after trim:demo_trim!
```

Yêu cầu:

Câu 1: Dựa trên bài thực hành tuần trước, hãy phân tích tệp Utils.java để

- Áp dụng các phép biến đổi chuỗi, hãy viết phương thức "List<String> getAllFunctions(File path)" để lấy toàn bộ phương thức static trong tệp đó qua các phép xử lý chuỗi, trong đó path là đường dẫn đến tệp Utils.java. Mỗi một phần tử String trong đối tượng trả về lưu mã nguồn phương thức static tìm được.
- Viết phương thức "public String findFunctionByName(String name)" để tìm kiếm một phương thức trong tệp. Trong đó, name là tên của phương thức cùng với kiểu biến. Ví

dụ, phương thức “public static File findFileByName(String folderPath, String fileName)” có tên là “findFileByName(String,String)”.

Chú ý trường hợp phương thức đặt trong comment.

Các phương thức đặt trong lớp Week10 tuân theo code mẫu tại địa chỉ như sau:

<https://github.com/oopuet/Exercises/tree/master/Week10>

Tuần 11. Generic – Lập trình tổng quát

Một ví dụ đơn giản để có thể giúp ta dễ tiếp cận lập trình tổng quát trong Java. Sử dụng lập trình tổng quát trong Java, ta có thể viết một phương thức chung để xếp thứ tự mảng của đối tượng, sau đó gọi phương thức generic này với các mảng integer, các mảng double, các mảng String,... để xếp thứ tự các phần tử của mảng.

❖ Phương thức tổng quát – Generic method

Ta có thể viết một khai báo phương thức tổng quát mà có thể được gọi với các tham số của các kiểu khác nhau. Dựa trên các kiểu tham số được truyền tới phương thức tổng quát này, bộ biên dịch xử lý mỗi lần gọi phương thức một cách thích hợp. Khi khai báo phương thức tổng quát, luôn cần kèm theo 1 đoạn tham số kiểu được giới hạn bởi các dấu ngoặc nhọn (< và >) mà đứng trước kiểu trả về của phương thức (<T> trong ví dụ dưới đây).

Ví dụ dưới đây minh họa cách ta có thể in mảng các kiểu khác nhau bởi sử dụng một phương thức tổng quát:

```
public class GenericMethodTest {
    public static <T> void printArr(T[] inputArray) {
        for ( T element : inputArray ) {
            System.out.printf( "%s ", element );
        }
    }

    public static void main( String args[] ) {
        Integer[] intArr = { 10, 12, 3, -4, 5 };
        Double[] doubleArr = { 5.1, 2.5, 7.3, -4.5 };
        Character[] charArr = { 'U', 'E', 'T', 'V', 'N', 'U' };

        System.out.println("Mang intArr bao gom:");
        printArr(intArr);

        System.out.println("\nMang doubleArr bao gom:");
        printArr(doubleArr);

        System.out.println("\nMang charArr bao gom:");
        printArr(charArr);
    }
}
```

Kết quả in ra như sau:

```
Mang intArr bao gom:
10 12 3 -4 5
Mang doubleArr bao gom:
5.1 2.5 7.3 -4.5
Mang charArr bao gom:
U E T V N U
```

❖ Lớp tổng quát – Generic class

Việc khai báo lớp tổng quát giống như khai báo 1 lớp bình thường, ngoại trừ tên lớp được theo sau bởi một đoạn tham số kiểu. Giống như với phương thức tổng quát, đoạn tham số kiểu của một lớp tổng quát có thể có một hoặc nhiều tham số kiểu, chúng phân biệt nhau bởi dấu phẩy. Ví dụ dưới đây minh họa cách ta định nghĩa một lớp tổng quát trong Java:

```
public class Box<T> {
    private T t;
    public void add(T t) {
        this.t = t;
    }
    public T get() {
        return t;
    }

    public static void main(String[] args) {
        Box<Integer> integerBox = new Box<Integer>();
        Box<String> stringBox = new Box<String>();

        integerBox.add(new Integer(10));
        stringBox.add(new String("Hello World"));

        System.out.printf("Gia tri integer la: %d\n",
            integerBox.get());
        System.out.printf("Gia tri string la: %s", stringBox.get());
    }
}
```

Kết quả in ra:

```
Gia tri integer la: 10
Gia tri string la: Hello World
```

Yêu cầu: Áp dụng lập trình tổng quát, hãy:

- Hãy viết phương thức sắp xếp tăng dần các phần tử có kiểu **dữ liệu nguyên thủy** (trừ **boolean**) và **String**. *Chú ý: Tự cài đặt thuật toán sắp xếp.*

Mô tả phương thức như bảng sau:

Tên	Tham số	Kiểu trả về	Modifiers
sortGeneric	arr: List<T>	List<T>	public, static

- Tạo lớp **Person** ([POJO](#)) gồm các thuộc tính: **name:** *String* (tên), **age:** *int* (tuổi), **address:** *String* (địa chỉ). Hãy cài đặt lớp **Person** thích hợp để có thể sắp xếp được bằng phương thức **sortGeneric** đã tạo ở trên. Các đối tượng **Person** được “sắp xếp” dựa theo việc sắp xếp *tên*, nếu *tên* giống nhau thì dựa trên sắp xếp *tuổi*.

Ví dụ: Có 3 đối tượng **Person**

- Person1 (name="Nguyen A", age=22, address=...)
- Person2 (name="Nguyen A", age=20, address=...)

3. Person3 (name="Le B", age=20, address=...)

Thì thứ tự sắp xếp là: Person3 < Person2 < Person1

Tuần 12. Design pattern: Composite, Strategy

Design pattern là một kỹ thuật trong lập trình hướng đối tượng, nó rất quan trọng khi giải quyết vấn đề của nhiều bài toán khác nhau. Design pattern là sự đúc kết kinh nghiệm để linh hoạt trong quá trình sử dụng về sau, yêu cầu mỗi lập trình viên muốn giỏi đều phải biết.

Về cơ bản, có 3 nhóm design pattern:

- Creational Design Pattern
- Structural Design Pattern
- Behavioral Design Pattern

❖ Singleton

Singleton pattern (thuộc Creational) đảm bảo chỉ duy nhất một new instance được tạo ra cho 1 lớp và nó sẽ cung cấp cho bạn một method để truy cập đến đối tượng duy nhất đó. Dù cho việc thực hiện cài đặt Singleton bằng cách nào đi nữa cũng đều dựa vào nguyên tắc dưới đây.

- private constructor để hạn chế khởi tạo đối tượng từ bên ngoài
- đặt private static variable cho đối tượng được khởi tạo, đảm bảo biến chỉ được khởi tạo trong chính lớp này.
- có một method public để return instance đã được khởi tạo ở trên.

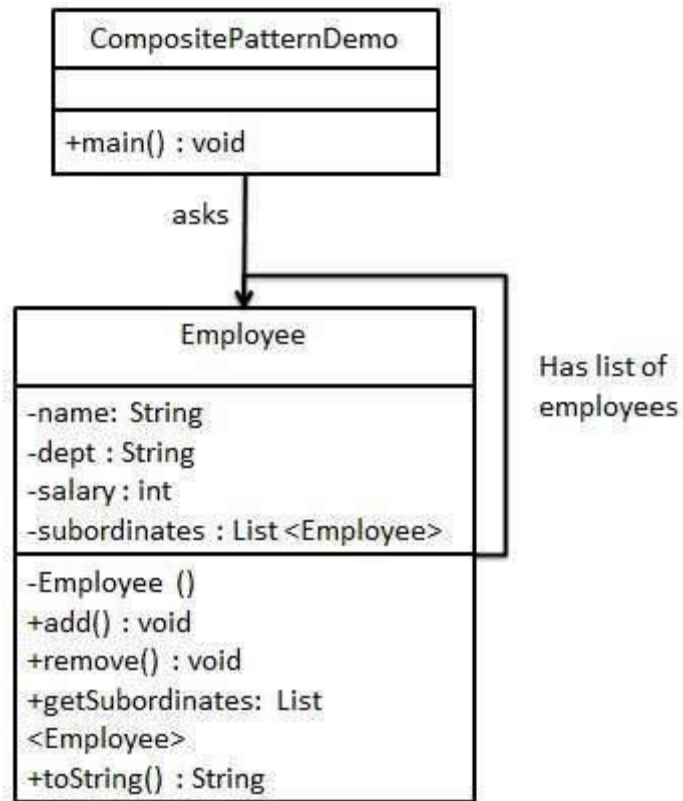
Code cài đặt cho Singleton đơn giản như dưới đây:

```
public class Singleton {  
  
    private static Singleton singleton = new Singleton( );  
  
    // Private Constructor để không cho phép  
    // bất cứ lớp nào khác khởi tạo  
    private Singleton() { }  
  
    // Phương thức static 'instance'  
    public static Singleton getInstance( ) {  
        return singleton;  
    }  
}
```

❖ Composite

Composite pattern (thuộc Structural) cho phép tương tác với tất cả các đối tượng tương tự nhau giống như là các đối tượng đơn hoặc collections. Ví dụ: Đối tượng File sẽ là 1 đối tượng đơn nếu bên trong nó không có file nào khác, nhưng đối tượng file (folder) sẽ được đối xử giống như 1 collections nếu bên trong nó lại có những File khác.

Trong ví dụ thể hiện ở biểu đồ lớp dưới đây, lớp **Employee** có 1 thuộc tính để lưu các cấp dưới của mình subordinates là 1 List<> các đối tượng **Employee**.



```

import java.util.ArrayList;
import java.util.List;

public class Employee {
    private String name;
    private String dept;
    private int salary;
    private List<Employee> subordinates;

    // constructor
    public Employee(String name,String dept, int sal) {
        this.name = name;
        this.dept = dept;
        this.salary = sal;
        subordinates = new ArrayList<Employee>();
    }

    public void add(Employee e) {
        subordinates.add(e);
    }

    public void remove(Employee e) {
        subordinates.remove(e);
    }

    public List<Employee> getSubordinates(){
  
```

```

        return subordinates;
    }

    public String toString(){
        return ("Employee :[ Name : " + name + ", dept : " + dept + ",
salary : " + salary+" ]");
    }
}

public class CompositePatternDemo {
    public static void main(String[] args) {

        Employee CEO = new Employee("John","CEO", 30000);

        Employee headSales = new Employee("Robert","Head Sales",
20000);

        Employee headMarketing = new Employee("Michel","Head
Marketing", 20000);

        Employee clerk1 = new Employee("Laura","Marketing", 10000);
        Employee clerk2 = new Employee("Bob","Marketing", 10000);

        Employee salesExecutive1 = new Employee("Richard","Sales",
10000);
        Employee salesExecutive2 = new Employee("Rob","Sales", 10000);

        CEO.add(headSales);
        CEO.add(headMarketing);

        headSales.add(salesExecutive1);
        headSales.add(salesExecutive2);

        headMarketing.add(clerk1);
        headMarketing.add(clerk2);

        //print all employees of the organization
        System.out.println(CEO);

        for (Employee headEmployee : CEO.getSubordinates()) {
            System.out.println(headEmployee);

            for (Employee employee : headEmployee.getSubordinates()) {
                System.out.println(employee);
            }
        }
    }
}

```

Kết quả in ra như sau:

```

Employee :[ Name : John, dept : CEO, salary :30000 ]
Employee :[ Name : Robert, dept : Head Sales, salary :20000 ]
Employee :[ Name : Richard, dept : Sales, salary :10000 ]

```

```
Employee :[ Name : Rob, dept : Sales, salary :10000 ]
Employee :[ Name : Michel, dept : Head Marketing, salary :20000 ]
Employee :[ Name : Laura, dept : Marketing, salary :10000 ]
Employee :[ Name : Bob, dept : Marketing, salary :10000 ]
```

❖ Strategy

Khi áp dụng strategy pattern (thuộc Behavioral) thì các hành vi hoặc giải thuật của một class có thể thay đổi ở runtime. Strategy pattern được sử dụng khi có nhiều lớp chỉ khác nhau về hành vi, khi cần các biến thể khác nhau của thuật toán. Strategy pattern định nghĩa mỗi hành vi trong lớp riêng, loại bỏ sự cần thiết cho các câu lệnh có điều kiện. Nó giúp dễ dàng mở rộng và kết hợp hành vi mới mà không thay đổi ứng dụng.

```
public interface Strategy {
    int doOperation(int num1, int num2);
}

public class OperationAdd implements Strategy{
    @Override
    public int doOperation(int num1, int num2) {
        return num1 + num2;
    }
}

public class OperationSubstract implements Strategy{
    @Override
    public int doOperation(int num1, int num2) {
        return num1 - num2;
    }
}

public class Context {
    private Strategy strategy;

    public Context(Strategy strategy){
        this.strategy = strategy;
    }

    public int executeStrategy(int num1, int num2){
        return strategy.doOperation(num1, num2);
    }
}

public class StrategyPatternDemo {
    public static void main(String[] args) {
        Context context = new Context(new OperationAdd());
        System.out.println("10 + 5 = " +
context.executeStrategy(10, 5)); // 15

        context = new Context(new OperationSubstract());
        System.out.println("10 - 5 = " +
context.executeStrategy(10, 5)); // 5
    }
}
```

Yêu cầu:

Câu 1. Hãy tìm mẫu thiết kế thích hợp với cấu trúc dữ liệu cho bài toán phả hệ như sau:

- Từng cá nhân cần lưu các thông tin gồm: ngày tháng năm sinh, giới tính, v.v.
- Một cá nhân có thể có một nhiều con hoặc không có con
- Một cá nhân có thể kết hôn hoặc không

Ví dụ: James kết hôn với Hana sinh ra hai người con là Ryan và Kai. Ryan không lấy vợ. Kai lấy Jennifer sinh ra bốn người con gồm hai nam, hai nữ. Bốn người con này tiếp tục kết hôn và sinh con đẻ cái.

Dựa trên mẫu thiết kế vừa tạo, hãy:

- Tìm tất cả các cá nhân không kết hôn trong danh sách phả hệ
- Tìm tất cả các cặp vợ chồng có hai con trong danh sách phả hệ
- Tìm tất cả các thế hệ mới nhất trong danh sách phả hệ
- Viết phương thức main kiểm tra

Gợi ý: Sử dụng mẫu thiết kế Composite vì mẫu này thích hợp để lưu danh sách dạng cây

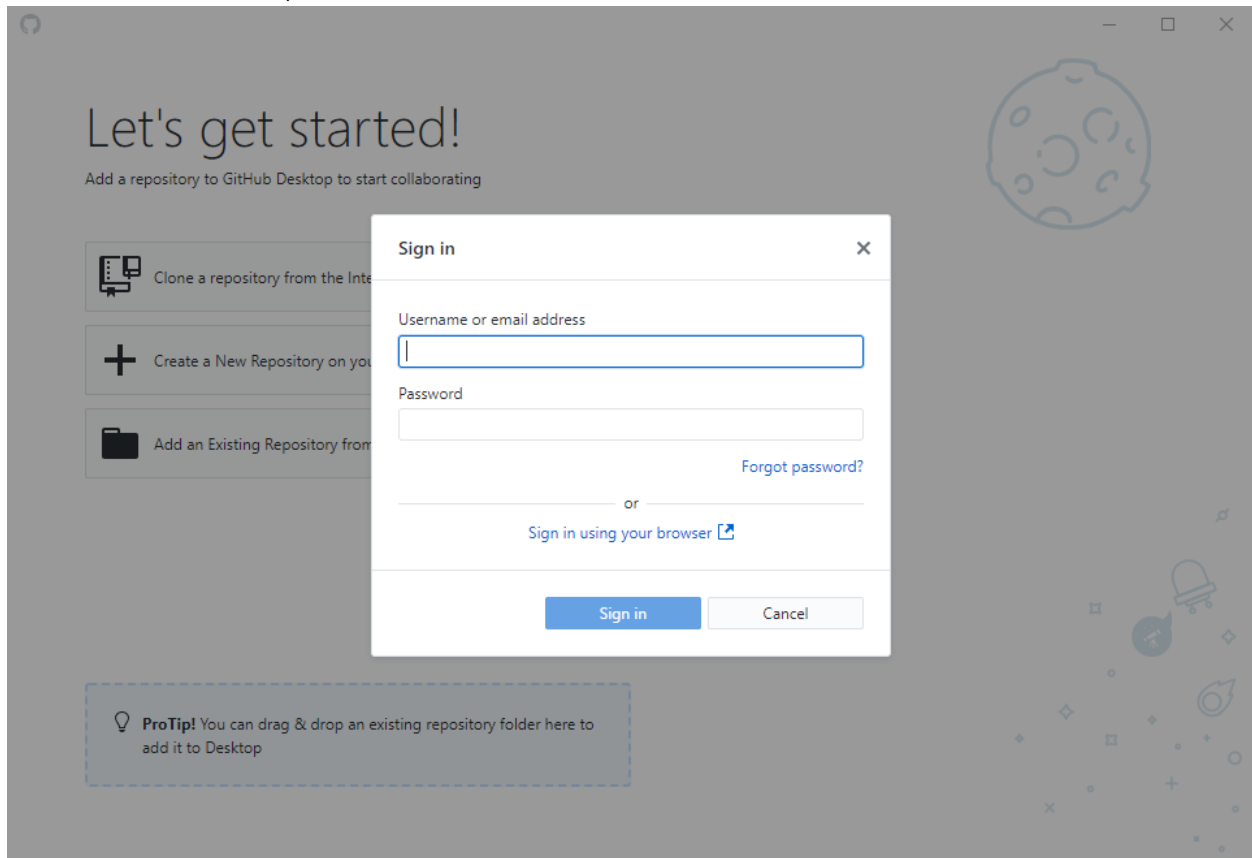
Câu 2. Cho một mảng các số nguyên có thể được sắp xếp tăng dần, hoặc giảm dần sử dụng thuật toán sắp xếp nổi bọt, hoặc sắp xếp chọn. Sau này, James mong muốn áp dụng thêm các thuật toán sắp xếp khác (hiện tại James chưa muốn cài đặt). Hãy giúp James xây dựng mẫu thiết kế thích hợp nhất. Viết phương thức main để chạy thử chương trình.

Gợi ý: Sử dụng mẫu thiết kế Strategy

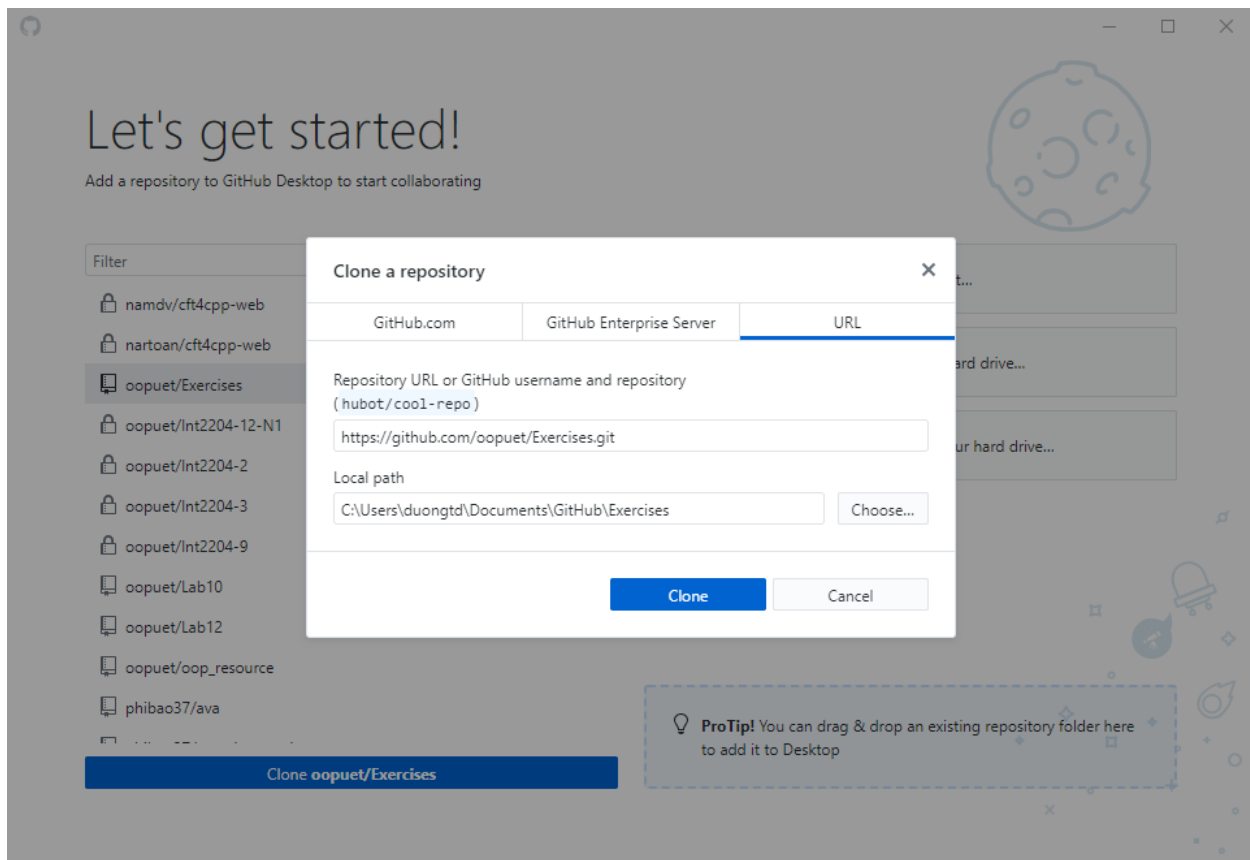
Tuần phụ 1. Cơ bản về Git và hướng dẫn sử dụng Github Desktop

Hướng dẫn này nhằm hướng tới việc giúp bạn có thể tạo mới 1 repo trên 1 git server như Github/Gitlab, clone repo về local, pull/push code giữa local và server.

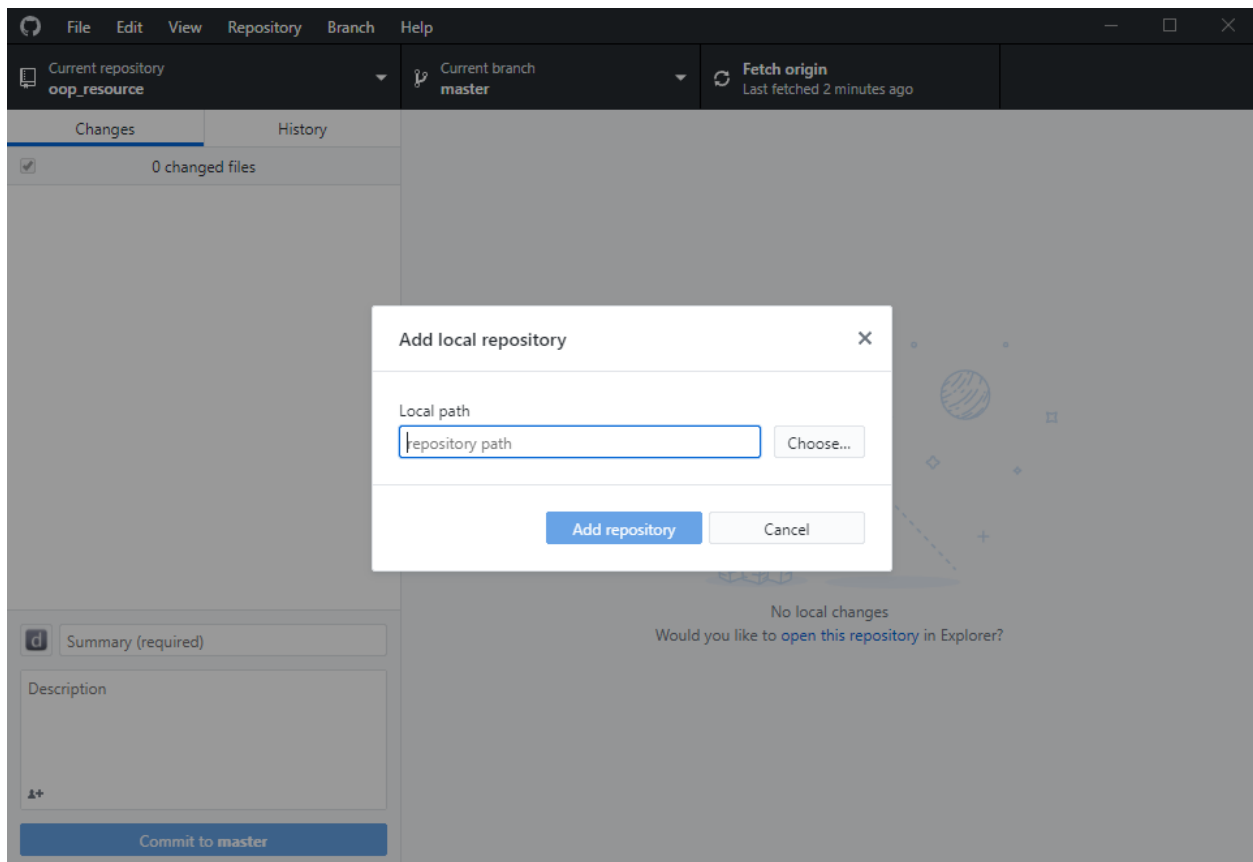
1. Đăng ký 1 tài khoản trên [Github](#) (có thể dùng [Gitlab](#) cũng tương tự), nếu có rồi thì bỏ qua bước này.
2. Tải và cài đặt Github Desktop tại [đây](#)
3. Cài đặt hoàn tất, mở Github Desktop, click chọn “Clone a repository from the Internet”, click chọn “Sign in”, sau đó đăng nhập theo tài khoản Github của mình (đã đăng ký tại bước đầu tiên).



4. Sau đó chọn đúng repo mình muốn clone về local, chọn nơi sẽ lưu repo tại local, và click “Clone”.



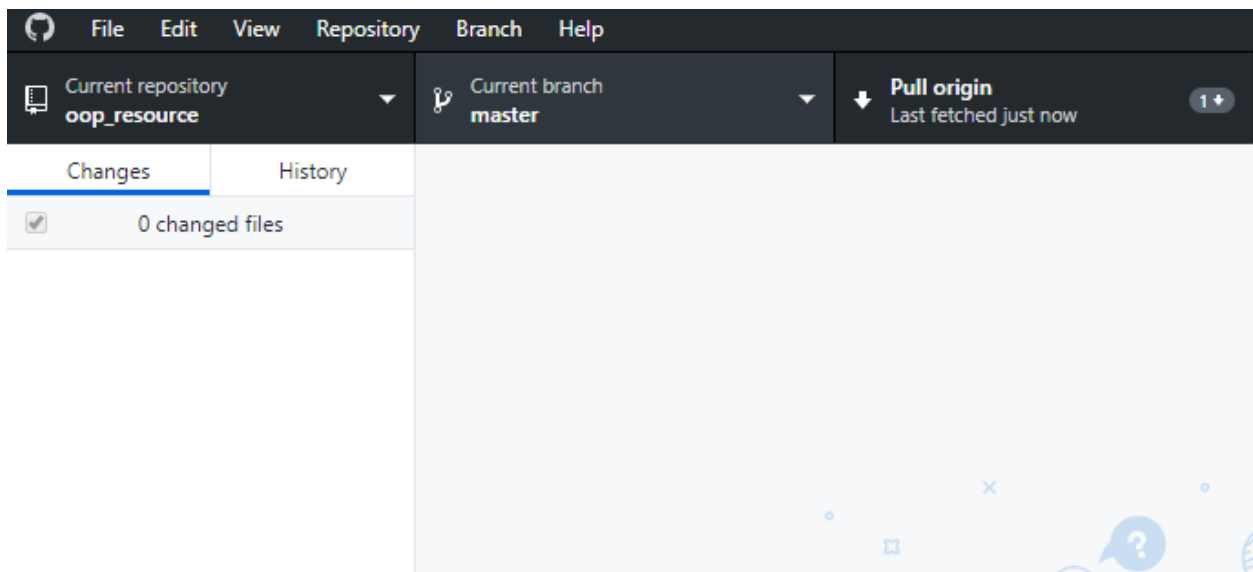
- Trong trường hợp bạn đã clone repo từ trước đó (sử dụng cmd or others), chọn File -> Add local repository, chọn tới folder chứa repo tại local (folder chứa thư mục .git).



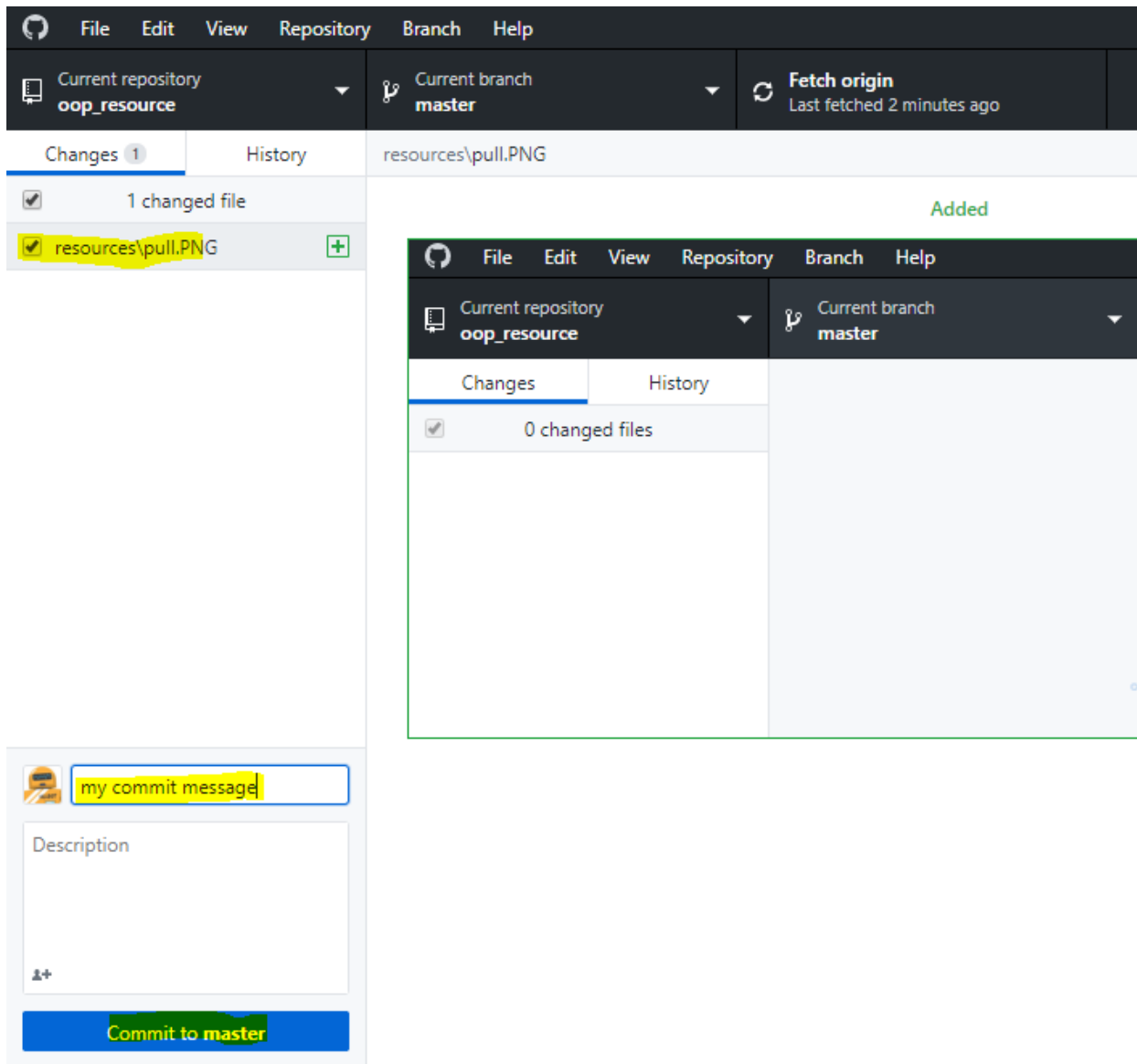
6. Với các lần sau:

- Đảm bảo đã chọn đúng repo, tại trên cùng bên trái 'Current repository'

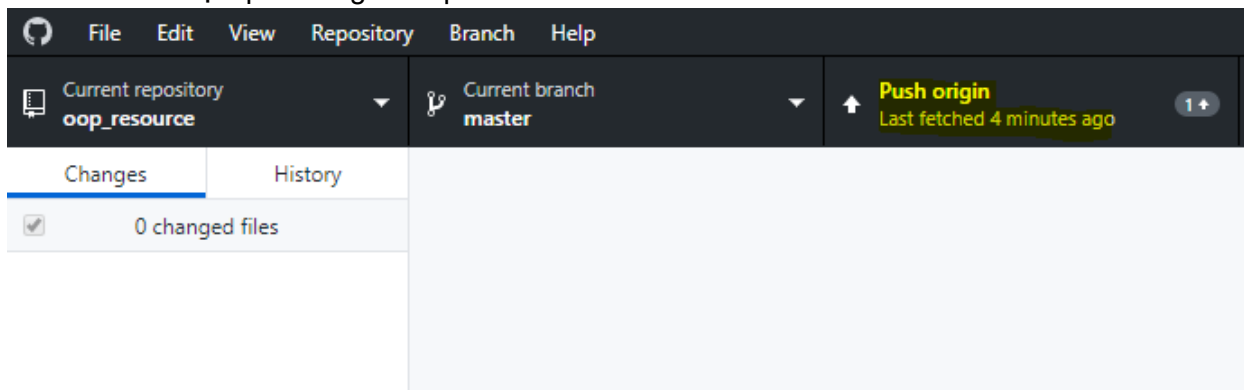
Chọn 'Fetch origin' tại bên phải trên cùng, nếu có xuất hiện 'Pull origin', click để pull change từ server về.



- Sau khi đã thay đổi mã nguồn và mong muốn commit lên server, bạn hãy chọn các file thay đổi (bỏ tick những file không push lên, ví dụ như file .class). Thêm commit message, chọn 'commit to master' ở dưới.



- click chọn 'push origin' để push lên server



- Cuối cùng lên <http://github.com> để kiểm tra commit của mình.

Tuần phụ 2. Cơ bản về đồ hoạ trong Java

Phần I: Làm quen với Java GUI (Swing)

Trong phần này, bạn sẽ được làm quen với các thành phần cơ bản của Java Swing – là một trong những nhóm thành phần thuộc **JFC** (Java Foundation Classes) cung cấp các thành phần giao diện người dùng cho ứng dụng Java.

Các thành phần của Swing được chia ra ba loại:

A, Container: **JFrame**, **JApplet**, **JDialog**

B, JComponent: **JPanel**, **JButton**, **JCheckBox**, **JRadioButton**, **JSlider**,...

C, Text components: **JLabel**, **TextField**, **TextArea**,...

- Ví dụ dưới đây giúp bạn tìm hiểu về **JFrame**, **JLabel**.

```
// 1. Tạo frame với JFrame
JFrame frame = new JFrame();

// 2. Đặt sự kiện tắt ứng dụng khi đóng frame
// Bạn có thể thử đặt HIDE_ON_CLOSE, DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE để hiểu thêm
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

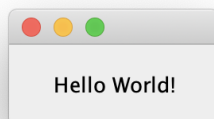
// 3. Tạo một label để gắn vào frame
JLabel label = new JLabel("Hello World!");
label.setHorizontalAlignment(JLabel.CENTER);
label.setVerticalAlignment(JLabel.CENTER);

// 4. Gắn label vào frame tại vị trí ở giữa
// Mặc định container sử dụng BorderLayout
// Xem thêm các loại layout ở đây: https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html
frame.getContentPane().add(label, BorderLayout.CENTER);

// 4. Đặt kích thước frame dựa theo kích thước các thành phần bên trong, cụ thể ở đây là label
frame.pack();

// 5. Hiện frame
frame.setVisible(true);
```

Kết quả là một cửa sổ như dưới đây:

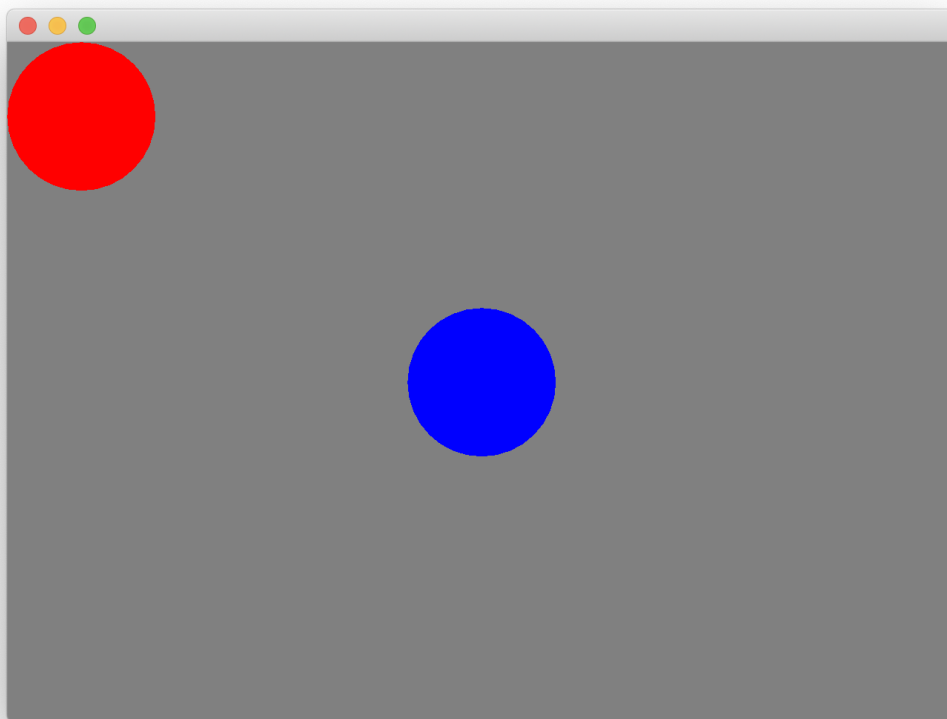


Phần II: Làm quen với Java2D API

Java2D cũng là một nhóm thành phần thuộc **JFC**, cho phép bạn tạo các thành phần đồ họa 2D theo ý muốn. Cụ thể Java2D cho phép bạn:

- Vẽ đường thẳng và bất kì dạng hình học mong muốn (hình tròn, hình vuông, hình elíp,...)
- Tô các hình này với màu sắc hoặc texture mong muốn
- Chèn text, hình ảnh
- Cho phép di chuyển, phóng to, xoay,... các đối tượng trong quá trình rendering liên tục

Ví dụ dưới đây minh họa cách sử dụng một số API cơ bản của Java2D. Chương trình này vẽ một cửa sổ có nền màu xám, và hai hình tròn màu đỏ và xanh ở phía trên.



Dưới đây là mã nguồn và comment giải thích của ví dụ trên:

```
import javax.swing.*;
import java.awt.*;
```

```

/**
 * 1. Tạo lớp Java2DPanel kế thừa từ JPanel.
 * Mục đích để ta có thể tự vẽ được các thành phần đồ họa theo ý muốn (custom graphics)
 */
class Java2DPanel extends JPanel {

    /**
     * 2. Ghi đè phương thức này giúp ta có đối tượng Graphics từ Java2D
     * Với đối tượng Graphics này, ta có thể tùy thích tạo ra bất cứ thứ gì mong muốn: hình tròn, elíp, đa giác, chèn
     ảnh, text,...
     * Xem thêm thông tin ở đây: https://docs.oracle.com/javase/8/docs/api/java/awt/Graphics.html
     */
    @Override
    protected void paintComponent(Graphics g) {
        // 3. Tô hình chữ nhật có kích thước bằng kích thước của panel tại vị trí (0, 0) với màu xám
        // Mục đích: muốn đặt màu nền cho JPanel màu xám
        // Chú ý trong trường hợp này, kích thước của panel bằng kích thước frame bên ngoài
        g.setColor(Color.GRAY);
        g.fillRect(0, 0, this.getBounds().width, this.getBounds().height);

        /**
         * Đối tượng nào được vẽ sau thì sẽ đè lên hình được vẽ trước,
         * vì thế các hình tròn được vẽ khi đã vẽ màu nền cho panel
         */

        // 4. Vẽ một hình tròn màu đỏ, có đường kính 100 đơn vị tại vị trí (0, 0)
        g.setColor(Color.RED);
        g.fillOval(0, 0, 100, 100);

        // 5. Vẽ một hình tròn màu xanh, có đường kính 100 đơn vị tại vị trí chính giữa panel
        g.setColor(Color.BLUE);
        g.fillOval((this.getBounds().width - 100) / 2, (this.getBounds().height - 100) / 2, 100, 100);
    }
}

/**
 * 6. Tạo lớp Java2D kế thừa JFrame, làm container chứa panel
 */
class Java2D extends JFrame {

    public static final int WIDTH = 640;
    public static final int HEIGHT = 480;

    // 7. Tạo đối tượng panel
    private JPanel panel = new Java2DPanel();

    public Java2D() {
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);

        // 8. Đặt kích thước ban đầu cho frame là 640x480
        this.setPreferredSize(new Dimension(WIDTH, HEIGHT));

        // 9. Gắn panel vào frame
        this.add(this.panel, BorderLayout.CENTER);

        this.pack();
        this.setVisible(true);
    }

    public static void main(String[] args) {

```

```
    new Java2D();  
}
```

II. Bài tập lớn

Đề 1. Phân tích chương trình

Hãy viết chương trình phân tích dự án viết bằng Java Standard Edition (SE) có giao diện đồ họa bằng Java Swing hoặc các công nghệ khác dựa trên Java. Chương trình bao gồm hai chức năng chính như sau:

- + Phân tích mã nguồn của một dự án Java SE lưu thành kiểu dữ liệu có cấu trúc trong bộ nhớ.
- + Hiển thị biểu đồ lớp (class diagram) mô tả quan hệ extends, implements,... của các lớp trong dự án. Biểu đồ này có khả năng tương tác với các chức năng cơ bản (nhưng không giới hạn) như sau: phóng to/thu nhỏ, kéo thả, sắp xếp bằng tay các thành phần trong biểu đồ, tự động sắp xếp các thành phần trong biểu đồ.

Điểm bài tập này sẽ được chấm hai lần dựa theo danh sách các bài kiểm tra sau:

Bài kiểm tra	Điểm
Đọc và phân tích được tên một lớp	0,5
Hiển thị được một lớp chỉ gồm tên trên biểu đồ	0,5
Phân tích được các thuộc tính và phương thức trong một lớp	1
Hiển thị đầy đủ thông tin của một lớp trên biểu đồ	0,5
Phân tích được toàn bộ các lớp	1
Hiển thị được toàn bộ các lớp	0,5
Phân tích được mối quan hệ ở mức lớp bao gồm: kế thừa (generalization) và cài đặt/hiện thực (realization / implementation)	1
Hiển thị được các mối quan hệ ở mức lớp lên biểu đồ	1,5

Phân tích được mối quan hệ ở mức thực thể là association	1
Chức năng phóng to thu nhỏ biểu đồ	0,5
Chức năng kéo thả trong biểu đồ	0,5
Chức năng tự động sắp xếp biểu đồ	0,5
Chức năng xuất biểu đồ ra ảnh raster (jpeg, png, ...) / ảnh vector (svg)	0,5
Thẩm mỹ của mã nguồn và giao diện	0,5

Tham khảo:

- https://en.wikipedia.org/wiki/Class_diagram

- Java tutorial <https://docs.oracle.com/javase/tutorial/uiswing/start/index.html>

Đề 2. Tính giá trị biểu thức

Viết chương trình Java Swing tính giá trị biểu thức. Các phép toán trong biểu thức gồm +, -, *, /, % (phép lấy phần dư), round (phép làm tròn), sqrt (phép lấy căn bậc 2), ^n (phép lấy mũ bậc n).

Ngoài ra, bạn hãy viết test case để kiểm tra tính đúng đắn chương trình (dùng JUnit)

Đề 3. Chương trình học tiếng anh

Viết chương trình Java Swing giúp học từ vựng tiếng anh. Người dùng có thể tạo các bộ từ vựng (collection) với các chủ đề khác nhau để học tiếng anh (chủ đề Du lịch, sức khỏe, v.v.). Các chức năng chính của chương trình gồm:

- + Chức năng ôn tập từ vựng thuộc về một chủ đề dựa trên thuật toán bạn định nghĩa (tham khảo thuật toán ôn từ của Anki, Memrise)
- + Chức năng quản lý bộ từ vựng: thêm, xóa, sửa tên, ghép hai bộ từ vựng thành một bộ từ vựng mới
- + Chức năng quản lý danh sách từ vựng (thêm, xóa, sửa)
- + Mỗi ngày hiện n từ mới để học và ôn tập m từ lâu nhất chưa ôn
- + Thêm ảnh minh họa cho từ vựng ????? ***** (bỏ)

- + Kiểm tra việc học từ thông qua trắc nghiệm cả xuôi và ngược
- + Thống kê dữ liệu về việc học từ vựng
- + Tra cứu từ trong danh sách các từ đã học
- + Hỗ trợ nhập từ vựng từ tệp Excel; xuất danh sách từ vựng ra tệp Excel ***** (điểm thưởng)

Đề 4. Bài xếp hình

Đề 5. Game bomberman

Đề 6. Game minesweeper