OOP With Java - Final Exam

ENSAI 2A UE8

Ludovic Deneuville

2025-04-24

Instructions

- Paper-based exam
- Duration: 1h30
- 1 handwritten A4 sheet (front and back) allowed
- No calculator

! Before you start

The subject is composed of multiple pages.

- You can answer in English or French.
- The exercises can be addressed in the order of your choice.
- If you get stuck on a question, quickly move on to the next one.
- Unless otherwise stated, the questions expect short answers.
- Strictly follow instructions. If unclear, write your assumptions.
- Respect good coding and language practices.
- Answer the **MCQ** by copying the question number and chosen answers; no justification required.

1 Course questions (2 points)

> **Instructions**
>
> Give short answers to the following questions:

a. What is a constructor?

A constructor is a special method in a class that is called when an object of that class is instantiated.

b. What commands did you use to compile and run your Java code?

- `javac File.java`: Compiles the File.java source file into bytecode.
- `java MyClass`: Runs the compiled MyClass bytecode.
- `mvn compile`: Compiles the source code of a Maven project.
- `mvn exec:java`: Executes the main class of a Maven project.

c. Explain what is brute force?

Brute force is a problem-solving method that involves systematically checking all possible solutions until the correct one is found.

d. What is a mutation test?

A mutation test evaluates the effectiveness of your tests by introducing small changes (mutations) to your code and checking if your tests can detect these changes.

e. What is mapReduce and what are its stages?

MapReduce is a programming model for processing large datasets in a distributed computing environment. It breaks down complex tasks into simpler, parallel operations. The MapReduce workflow consists of three main stages:

- **Split**: Input data is divided into smaller chunks.
- **Map**: Each chunk is processed by a Map function.
- **Reduce**: Values are aggregated by a Reduce function.

2 MCQ (10 points)

> **Instructions**
>
> Each question can have one or more correct answers (at least one answer is correct).
> Correct answer: 1 point. Partially correct answer: ratio. If there is a wrong answer: 0.

2.1 What will happen if you try to use a local variable without initializing it?

a. It will be assigned a default value of 0.
b. It will be assigned a default value of null.
c. **It will cause a compilation error.**
d. The compiler will automatically initialize it for you.

2.2 What can be said about Java's type system?

a. **Java uses strong typing, as a variable cannot change its type once declared.**
b. Java uses dynamic typing, like Python.
c. **Java uses static typing, as types are checked at compile-time.**
d. Java allows changing the type of a variable at runtime.

2.3 Which characteristics describe Java as a programming language?

a. **Java is portable because it can run on any machine with a JVM installed.**
b. **Java is compiled to bytecode, allowing it to be run on different platforms.**
c. **Java follows the "Write Once, Run Anywhere" principle.**
d. Java does not have automatic memory management, which must be handled by developers.

2.4 What are the valid signatures of a `main` method in Java?

a. `public void main(String[] args)`
b. **`public static void main(String[] args)`**
c. `static public main(String args[])`
d. `void main(String[] args)`

2.5 Using the class below, what code is used to create a Person object?

```
public class Person {
    private String name;
    private int age;
    private boolean isStudent = true;

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

a. `Person p = Person ("Lisa", 30, false);`
b. `Person p = Person ("Lisa", 30);`
c. **`Person p = new Person ("Lisa" , 30);`**
d. `p = new Person ("Lisa", 30, true);`

2.6 Which data structure is most suitable for storing the days of the week?

single answer

a. **`Enum`**
b. `Set<String>`
c. `ArrayList<String>`
d. `Map<Integer, String>`

2.7 What does lazy evaluation mean in programming?

a. **Evaluating expressions only when their values are needed.**
b. Storing intermediate results to avoid recalculating them.
c. Always evaluating all expressions at the start of the program.
d. Automatically optimizing code to make it run faster.

2.8 Which file is essential for any Maven project?

a. build.xml
b. application.yml
c. **pom.xml**
d. config.xml

2.9 Which statements are true about Apache Maven?

a. **It simplifies dependency management by automatically downloading required libraries.**
b. **It enforces a standardized project structure.**
c. It is primarily a code editor with built-in compilation features.
d. **It provides a standardized build lifecycle.**
e. It requires manual configuration of every dependency.

2.10 Which layer does this MysteryClass belong to?

```
public class MysteryClass {
    @Autowired
    private AthleteService athleteService;

    @GetMapping("/api/athletes")
    public ResponseEntity<List<Athlete>> getAllAthletes() {
        List<Athlete> athletes = athleteService.findAll();
        return ResponseEntity.ok(athletes);
    }
}
```

a. **Controller**
b. Repository
c. Model
d. Service

2.11 Which are key features of the Spring Framework?

a. **Dependency Injection**
b. **Inversion of Control**
c. **Aspect-Oriented Programming**
d. Garbage Collection

3 Java Basics (4 points)

a. Write a function `isPositive` in Java that takes an integer and returns true if positive or zero, false otherwise.

```java
public static boolean isPositive(int number) {
    return number >= 0;
}
```

b. Write a Java function that prints multiplication tables from 1×1 to 9×9 in this format:

```
1 x 1 = 1
1 x 2 = 2
...
1 x 9 = 9
-----------
2 x 1 = 2
...
9 x 9 = 81
```

```java
public static void printMultiplicationTables() {
    for (int i = 1; i <= 9; i++) {
        for (int j = 1; j <= 9; j++) {
            System.out.println(i + " x " + j + " = " + (i * j));
        }
        if (i < 9) {
            System.out.println("-----------");
        }
    }
}
```

c. What will the code below display?

```java
int i = 0;
while (i < 10) {
    i++;
    if (i == 2) {
        continue;
    } else if (i == 5) {
        break;
    }
    System.out.println(i);
}
```

```
1
3
4
```

d. Give a short example of code using a Lombok annotation and explain its usefulness.

```java
@AllArgsConstructor
public class Person {

    @Getter
    private String name;
    private int age;
    private boolean isStudent = true;
}
```

With `@AllArgsConstructor`, Lombok generates this constructor for you, reducing boilerplate code.

With annotation `@Getter`, Lombok generates this getter method for you, making the code cleaner and more concise.

4 Java Code (4 points)

```java
public class Fraction {
    private int numerator;
    private int denominator;

    public Fraction(int numerator, int denominator) {
        if (denominator == 0) {
            throw new IllegalArgumentException("Denominator cannot be zero.");
        }
        this.numerator = numerator;
        this.denominator = denominator;
    }
}
```

a. Can the code below work outside the Fraction class? If not, suggest a solution.

```java
Fraction f = new Fraction(1, 2);
System.out.println(f.denominator);
```

The code provided will not work outside the Fraction class because the denominator attribute is private and cannot be accessed directly. To solve this, you can add a public getter method to the Fraction class to access the denominator.

b. Override the `toString` method so it returns the fraction in the format "numerator/denominator".

```java
    @Override
    public String toString() {
        return this.numerator + "/" + this.denominator;
    }
```

c. Add a public method `inverse` that returns a new Fraction with the numerator and denominator swapped. Write JavaDoc.

```java
    /**
     * Returns a new Fraction that is the inverse of this fraction,
     * with the numerator and denominator swapped.
     *
     * @return a new Fraction with the numerator and denominator swapped
     * @throws ArithmeticException if the numerator is zero
     */
    public Fraction inverse() {
        if (this.numerator == 0) {
            throw new ArithmeticException("Numerator cannot be zero for inverse operation.")
        }
        return new Fraction(this.denominator, this.numerator);
    }
```

d. Write unit tests for `inverse` using JUnit.

To check exceptions: `assertThrows(ExpectedException.class, () -> {<Code>});`

```java
@Test
public void testInverseOk() {
    Fraction f1 = new Fraction(1, 2);
    Fraction inverseF1 = f1.inverse();
    assertEquals("2/1", inverseF1.toString());

@Test
public void testInverseWithZeroNumerator() {
    Fraction f = new Fraction(0, 2);
    assertThrows(ArithmeticException.class, () -> {
        f.inverse();
    });
}
```

e. Give git commands to push your code to the remote repository.

```
$ git add Fraction.java
$ git add FractionTest.java
$ git commit -m "add inverse method and tu"
$ git push
```