Object-Oriented Programming
June 2024

Notes or documents of any kind forbidden. Duration: 3 1/2h. Please answer the questions on separate sheets labeled with your name, section, and student ID.

---

1. The problem consists in programming in Java a class `Dice` for an web application implementing a board game. This class must have the following features:

   - Instantiating `Dice` simulates a dice roll by randomly selecting an integer between 1 and 6. This operation produces an immutable object that represents the result of the roll. It must be possible to query this result by invoking the method `int getValue()`. Dice rolls must be fair, i.e., the probability of occurrence must be equal for all values.
   - The class maintains statistics about the number of dice rolls and the number of occurrences observed for each value. A class method `double getFrequency(int n)` must return, for a given dice value `n`, the proportion of dice rolls that have produced this value. For instance, if there have been 10 dice rolls, among which 2 have produced the value 4, then `getFrequency(4)` will return 0.2.
   - Instances of `Dice` must be clonable, comparable to each other, and serializable.
   - In case of any error, a dedicated exception should be thrown.

   In order to generate random numbers, you are asked to use the class `SecureRandom` of the Java Class Library, which implements a cryptographically secure generator. This class belongs to the package `java.security`, and has the following documentation:

   - Before rolling any dice, a random number generator must be created by instantiating `SecureRandom` once, without any argument.
   - Calling `int nextInt(int k)` on the resulting object returns a freshly drawn random number, with a uniform probability distribution in the interval $[0, \mathtt{k} - 1]$.This method can be called any number of times.

   Note: You are free to implement any additional classes required by your solution, as well as to choose the interpretation of details that are not specified in this problem statement.

2. A role-playing game requires a special kind of dice called *D20*, that can produce a value between 1 and 20 instead of between 1 and 6.

   In an application that needs to handle both regular (6-sided) and D20 (20-sided) dice, would you define the class `D20Dice` suited for D20 dice as a subclass of `Dice`, or would you organize differently the hierarchy between the classes of your program? Please answer the question by drawing a diagram showing the subclassing relation that you would define between your classes, and explaining which application of inheritance is used in each branch of this relation.

   Notes:

   - You are not asked to program explicitly the class `D20Dice`.
   - If your solution requires to modify your answer to Problem 1, it is sufficient to explain how you would perform this modification, without describing it in detail.

3. (a) Describe all the constructors that are invoked during the evaluation of the Java expression `new String("OOP")`, as well as their arguments, and the order in which they are invoked.

   (b) Explain why the type erasure mechanism used by the Java implementation of generics does not allow to define arrays whose elements are defined with a generic type.

   (c) In Java, when is it possible for the lock of an object to be acquired multiple times? Why is this feature essential?

4. The problem consists in programming in Java a class `Flag` intended for a multithreaded piece of software. This class must satisfy the following requirements:

   - An instance of this class represents a flag that is at all times either *raised* or *lowered*.
   - Two methods `void raise()` and `void lower()` make it possible to (respectively) raise and lower a flag. Those methods can be invoked at any time from any thread.
   - A method `void waitUntilRaised()` can be invoked for blocking (as efficiently as possible) the calling thread until the flag is raised by another thread. If the flag is already raised, this method has no effect. When a flag is raised, all the threads waiting on that flag must be simultaneously unblocked.