Object-Oriented Programming
August 2019

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. A *binary tree* is a data structure composed of some number of *nodes*. Each tree contains a distinguished node called the *root*. Each node is either a *leaf*, or an *internal node*. Each leaf is associated with a *payload*, represented by an arbitrary object of a given type. Each internal node is characterized by its *left* and *right children*, which are nodes. Every node is the child of exactly one other node, except for the root that is not the child of any node. The *depth* of a tree is the largest integer d for which there exists a sequence $n_0, n_1, \ldots, n_d$ such that n_0 is the root, and for each $i \in [1, d]$, the node n_i is a child of the node n_{i-1}.

   You are asked to program in Java a class for representing a binary tree, that satisfies the following requirements:

   - The class should use a type parameter for specifying the type of the payload of leaves.
   - Once it has been assigned, the payload of a leaf cannot be modified.
   - It must be possible to instantiate a binary tree composed of a single node, with a given payload.
   - It must be possible to construct a binary tree T by combining two existing trees T_1 and T_2 sharing the same payload type. This operation consists in creating a new node that becomes the root of T, such that its left child is the root of T_1, and its right child is the root of T_2. The caller of this operation is responsible for ensuring that the sets of nodes of T_1 and T_2 are disjoint.
   - It must be possible to compute the number of nodes and the depth of a tree.
   - Instances of this class must be clonable, comparable to each other (two trees are considered to be equal if they have the same structure, and if the payloads of their leaves are pairwise equal), and serializable.
   - In case of any error, a dedicated exception should be thrown.

   Note: You are free to implement any additional class required by your solution.

2. (All answers should be thoroughly justified.)

   (a) What are polymorphic methods? When a class defines polymorphic methods, how does the compiler determine which one to invoke?

   (b) Write a fragment of Java code that creates a 2-dimensional array of integers containing 100 rows, in which the 1st, 3rd, 5th, ... rows contain 1000 columns each, and the 2nd, 4th, 6th, ... ones contain 500 columns. The elements of the array can be left uninitialized.

   (c) Explain the limited form of multiple inheritance allowed by the Java language.

   (d) Explain the principles of constructor chaining in Java.

   (e) In Java, when is it mandatory for a method to define which exceptions it can trigger?

   (f) In Java, what are the operations performed by the methods `notify()` and `notifyAll()`? How do they differ?