Object-Oriented Programming
June 2016

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

1. During the development of a piece of mathematical software, a programmer needs to define a class for representing *closed bounded intervals*. Such an interval is denoted $[a, b]$, where a and b are integers such that $a \leq b$, and corresponds to the set of all real numbers x such that $a \leq x \leq b$. Note that, according to this definition, a closed bounded interval cannot be empty. The defined class should make it possible to

   - create a new interval for given values of a and b.
   - compute the convex union of two intervals I_1 and I_2, which is the smallest closed bounded interval that contains all the numbers belonging to I_1 or I_2. (As an example, the convex union of $[5, 7]$ and $[2, 4]$ is the interval $[2, 7]$.)
   - compute the intersection of two intervals I_1 and I_2, which is the closed bounded interval that contains exactly all the numbers belonging to both I_1 and I_2. (As an example, the intersection of $[1, 3]$ and $[2, 7]$ is the interval $[2, 3]$.)
   - compute the number of integer values contained in an interval. (As an example, the interval $[2, 7]$ contains 6 values.)
   - check whether two intervals are equal (which means that they correspond exactly to the same set of numbers).
   - clone an existing closed bounded interval.

   (a) Write a suitable CRC card for the closed bounded interval class, as well as for any auxiliary class that it may require. *Note:* The details that are not specified by the problem statement can be freely chosen.

   (b) Implement the closed bounded interval class in Java, using appropriate language mechanisms. In particular, equality checking and cloning must be implemented in the standard way, and all errors must be reported using suitably defined custom exceptions.

   (c) Explain (with justifications) whether your solution to (b) is *thread-safe*, in other words, whether multiple threads of a concurrent program could safely interact with a common instance of your class. *Note:* If the answer is negative, you are not asked to modify your implementation in order to make it thread-safe.

2. Consider the following Java program.

```
interface Time
{
  void beat();
}

class Clock implements Time
{
  private String t = "tick";
  public void beat()
  {
    System.out.println(t);
  }
}

class NewClock extends Clock
{
  private String t = "tock";
  public void beat()
  {
    System.out.println(t);
  }
}

public class Test
{
  public static void main(String[] args)
  {
    Time c = new NewClock();
    c.beat();
  }
}
```

(a) Does this program compile without reporting errors?

(b) If no, how can it be corrected? If yes, what does it display when it is run?

(Justify thoroughly your answers to both points.)

3. Consider the following partial Java implementation of a binary tree storing integers. (Only some methods related to left-hand nodes are shown.)

```
public class BTNode
{
  private int element;
  private BTNode left = null, right = null;
  ...
  protected BTNode(int element)
  {
    this.element = element;
  }
  protected BTNode getLeft()
  {
    return left;
  }
  protected void setLeft(BTNode node)
  {
    left = node;
  }
  ...
}

public class BTTree
{
  private BTNode root = null;
  private int size = 0;
  ...
  public BTNode getRoot()
  {
    return root;
  }
  public BTNode createRoot(int content)
  {
    root = new BTNode(content);
    size = 1;
    return root;
  }
  public static BTNode getLeft(BTNode node)
  {
    return node.getLeft();
  }

  // (continued on next page)
  public BTNode insertLeft(BTNode node, int content)
  {
    BTNode left = new BTNode(content);
    node.setLeft(left);
    ++size;
    return left;
  }
  ...
}
```

(a) Using Java generics, transform this code (by writing a new version of the classes `BTNode` and `BTTree`) in order to make this binary tree able to store objects of any type, instead of only integers.

(b) How would you modify your solution to (a) in order to obtain a tree that would only be able to store numbers (possibly of different types)? What would be an advantage of imposing such a restriction?