PE Questions for
GUIs, Networking and Multithreaded Programming

Medialogy, Semester 4, 2009
Aalborg University (Aalborg)

Multi-threaded programming

1. Give an example of a multi-threaded program and describe some of the tasks that are carried out by different threads in the program.
2. Did you use multi-threaded programming in your project? If so, how and why did you use it?
3. Explain what is meant by the term *race condition*.
4. Describe how new threads are created, started and stopped in Java.
5. There are two basic ways of creating threads in Java: one involves extending a superclass, the other involves implementing an interface. Briefly describe the two methods. Which is generally preferable and why?
6. The Thread class has *two* relationships with the Runnable interface. Draw a UML class diagram that shows the relationship between the Thread class and the Runnable interface.
7. Explain the meanings of the terms *interference* and *critical region* with regards to a multi-threaded program.
8. Explain how *synchronization* and *locks* can be used to avoid interference.
9. Why would you declare a method as being synchronized? What object does a synchronized method acquire a lock on?
10. Suppose I want to make sure that only the current thread accesses the object, `obj`, when I execute the following two lines:
    ```
    obj.methodA();
    obj2.methodB(obj);
    ```
    How can this be accomplished in Java?
11. Explain the difference between server-side and client-side synchronization. What are the advantages and disadvantages of each type of synchronization?
12. What static method is used to get the current thread in Java?
13. What static method is used to pause the current thread in Java? What kind of exception does this method throw?
14. In Java, how do you make the current thread wait until some other specified thread (call it *t*) has died?
15. Explain how a thread can be stopped in Java before it has finished executing. What happens when a thread is stopped by an instruction from some other thread?

16. Explain the function of the *interrupt status flag* in Java.
17. What method do we use in Java to determine if a thread, *t*, is still running?
18. What is a *guarded block*?
19. What does the term *busy waiting* mean with reference to a guarded block?
20. How can the wait and notifyAll methods be used to avoid busy waiting when creating guarded blocks in Java?

Graphical User Interfaces

21. What are the three different types of thread that are used in Swing? Briefly describe how each one is used.
22. Typically, what interface should a Java GUI implement and on what thread is it run?
23. How is a GUI typically started in Java?
24. What class of object is usually used to represent the main window of a Swing application?
25. What kind of thread should be used to run background tasks?
26. What thread must be used to run Swing event handling code?
27. What object in a JFrame contains most of the Swing components in the JFrame? What single type of Swing GUI component does this object not contain?
28. What JFrame method is used to calculate the size of the window it represents and format it before it is displayed?
29. What JFrame method is used to make a JFrame visible?
30. What kind of container do you use to hold Swing components in a region of a window in which the formatting is independent from the rest of the window?
31. Give two examples of a LayoutManager and describe how they control the formatting of components in a container.
32. How many Swing components can be placed in a single region of a window formatted using BorderLayout.
33. What is the default LayoutManager for a JFrame?
34. What is the default LayoutManager for a JPanel?
35. Explain the basic mechanism by which a user pressing a JButton can cause code to be executed.

Network programming

36. In networking, what is the *network interface layer*?
37. How many IP addresses are there? There are more than 1 billion computers in the world. Can you see a problem? (Bonus question – not covered in course: Do you know what is being done about it?)

38. What are the two main differences between TCP and UDP? Which is faster? Which is more reliable?
39. Give an example of an application where it might be preferable to use UDP instead of TCP
40. Explain the terms *socket* and *port*. How many different ports are there? How is each port identified? Can any port be used for any purpose?
41. Which standard Java package contains the class definitions that are used for network programming?
42. Consider the following URL:
    https://www.chromamorph.com:8080/papers/icmc.html#introduction
    Identify the protocol, host name, file name, port number, reference and resource name in this URL.
43. How do you create a Java URL object for the URL "http://www.smurf.com/splurge.html"?
44. Consider the following two lines of code:

```
URL yahoo = new URL("http://www.titanmusic.com/");
BufferedReader in = new BufferedReader(new
      InputStreamReader(yahoo.openStream()));
```

    What does the openStream() method provide? What does the InputStreamReader provide? What does the BufferedReader do?
45. Explain how a ServerSocket and Socket object can be used to establish a TCP connection between a server program and a client program.
46. Explain how a client program can send and receive datagrams to and from a server program using the InetAddress, DatagramPacket and DatagramSocket classes. In particular, you will have to use the send() and receive() methods of the DatagramSocket class and the getAddress() and getPort() methods of the DatagramPacket class.