1 Chapter 13

(1) ***What are three possible levels of concurrency in program?***
Answer: Three possible levels of concurrency in program are

- Machine instruction levels
- High-level language statement levels
- Program levels

(2) ***What is the difference between physical and logical concurrency?***
Answer: Physical concurrency will allow several programs to execute simultaneously on more than one processor while logical concurrency happens when the execution of several programs take place in an interleaving fashion on a single processor. In other words, physical concurrency uses the power of the hardware to execute many programs at the same time while logical concurrency uses the power of logics, coding to execute many programs at the same time.

(3) ***What is a thread of control in a program?***
Answer: Thread of control in a program is the sequence of program points reached as control flows through the program.

(4) ***What is a multithreaded program?***
Answer: A program that is designed to have more than one thread of control.

(5) ***What is a heavyweight task? What is a lightweight task?***
Answer:

- heavyweight task: are tasks that execute in their own address space
- lightweight task: are tasks that all run in the same address space (more efficient)

(6) ***What kind of tasks do not require any kind of synchronization?***
Answer: Disjoint task do not require an kind of synchronization because it does not communicate with or affect the execution of any other task in the program in any way.

(7) ***Specifically, what Java program unit can run concurrently with the main method in an application program?***
Answer: Java Threads

(8) ***Are Java threads lightweight or heavyweight tasks?***
Answer: Java threads are lightweight tasks.

(9) ***Problem set:***
Suppose two tasks A and B must use the shared variable Buf_Size. Task A adds 2 to Buf_Size, and task B subtracts 1 from it. Assume that such arithmetic operations are done by the three-step process of fetching the current value, performing the arithmetic, and putting the new value back. In the absence of competition synchronization, what sequences of events are possible and what values result from these operations? Assume the initial value of Buf_Size is 6. **Answer:**

```
1  Buf_Size = 6;
2  TaskA()
3  {
4    fetch();
5    Buf_Size += 2;
6    store();
7  }
8  TaskB()
9  {
10   fetch();
11   Buf_Size -= 1;
12   store();
13 }
```

There are 4 possible outputs for this program

(1) A finish first: $(6 + 2) - 1 = 7$

(2) Both A and B fetch at the same time

  (a) A puts the value back first: $6 - 1 = 5$

  (b) B puts the value back first: $6 + 2 = 8$

(3) B finish first: $(6 - 1) + 2 = 7$

2 Chapter 14:

1. ***What is the name of all C++ exception handlers?***
   Answer: In C++, all exception handlers are the catch functions in the try-catch blocks.

2. ***How can exceptions be explicitly raised in C++?***
   Answer: We can use the `throw` clause

```
if(endOfFile == true)
{
  throw new EndOfFileEx("this is the end of file");
}
```

3. ***How are exceptions bound to handlers in C++?***
   Answer:
   Exceptions is raised by the `throw [expression]` clause. The type of the `throw` expression will select the particular handler. Let's say the throw expression is type T, and a handler whose parameter is type T or any classes that is an ancestor of T matches will be chosen.
   The exception is also propagated from the inner `try` clause to the outer `try` clause, after that it will propagate to the caller of the function contains the `try` clause, then if it still doesn't find the handler, it will propagate to the function's caller. Finally, if the handler is still nowhere be found, the default handler will be called.

4. ***Does C++ include built-in exceptions?***
   Answer: No, there isn't built-in exceptions for C++.

5. ***What is the root class of all Java exception classes?***
   Answer: `Throwable` class

6. ***What is the parent class of most Java user-defined exception classes?***
   Answer: The `Exception` class

7. ***How can an exception handler be written in Java so that it handles any exception?***
   Answer: We will write a special handler that looks like this.

```
catch (Exception e)
{
  .......
}
```

   Because the Exception is the ancestor class, so it will always catch any exceptions that is raised.

8. ***What is the difference between checked and unchecked exceptions in Java?***
   Answer: Exceptions of class `Error` and `RuntimeException` and their descendants are called **unchecked exceptions**. All the other exceptions are called **checked exceptions**. Compiler and program don't handle **unchecked exceptions**, they only handle the **checked exceptions** with the throw clause or the handler within the method.

9. ***What is the purpose of the Java finally clause?***
Answer: The `finally` clause is used to execute the process that must be execute whether the `try` clause throws an exception and regardless of whether a thrown exception is caught in a method.

10.
```
class Big {
  int i;  float f;
  void fun1() throw int {
    ...
    try {
    ...
    throw i;     //#1
    ...
    throw f;     //#2
    ...
  }
  catch(float) {    }
  ...
  }
}
class Small {
  int j; float g;
  void fun2 () throw float {
    ...
    try {  ...
      try {
        Big.fun1();
        ....
        throw j;     //#3
        ...
        throw g;  //#4
        ...
      }
      catch(int) {...{
      ...
     }
    catch(float) {...}
  }
}
```

In each of the four throw statements, where is the exception handled? Note that fun1 is called from fun2 in class small.
Answer: -Throw i is handled in `catch(int)` in fun2 -Throw f is handled in `catch(float)` in fun1 -Throw j is handled in `catch(int)` in fun2 -Throw g is handled in `catch(float)` in fun2