Exercise 1

Object-oriented programming

8 Points

a) Name two possibilities to react to exceptions in Java. (2 points)

- Re-throw the exception, the method must declare the exception with `throws` (1 P)
- Catch (and handle) the exception with `try-catch` (1 P)

b) The following piece of code contains a logical mistake. Explain it and propose a solution. (2 points)

```java
public void doSomething() {
    try {
        someObject.process();
    } catch (Exception ex1) {
        throw new RuntimeException(ex1);
    } catch (IOException ex2) {
        System.err.println(ex2);
    }
}
```

The first `catch` block already catches all possible exception, the second `catch` block will therefore never be reached (1 P). Simply swap both `catch` to repair it (1 P).

c) Give a short and concise explanation for the concept of generic programming. Name two advantages of using generics in Java.

Generic programming allows you to created typed classes. The type parameter will be fixed to a concrete class (or interface) only upon usage of the generic class. The type parameter can then be used inside the class at arbitrary places such as method parameters or return values. (2 P).

Advantages are absence of explicit type casts (1 P) and type-safe collections (1 P).
Exercise 2 Parallel programming

15 Points

a) Name and describe two problems that can arise when executing two threads concurrently.

- data inconsistencies: two threads change the value of an object at the same time without synchronisation (2 P)
- deadlocks: two threads wait for each other (2 P)

b) Create a class `Locker`, which has two methods `lock` and `unlock`. Objects of this class are used to limit the number of threads that can enter a certain block of a program at the same time. The constructor of the class takes the maximum number of allowed threads. `lock` is called before the block is entered. If there are fewer threads than allowed already in the block, the current thread may enter the block. If not, it will be blocked and has to wait until another thread leaves the block. Upon leaving the block, `unlock` must be called. The following piece of code demonstrates the usage of the `Locker`:

```java
public class Test {
    private Locker locker = new Locker(3);

    public void doSomething() {
        locker.lock();
        try {
            // do something
        } finally {
            locker.unlock();
        }
    }
}
```

You must not use classes from the `java.util.concurrent` package (e.g. `ReentrantLock`). Instead you should use Java’s default synchronisation mechanisms. For simplification you must not for threads that call `unlock` without having called `lock` beforehand.
public class Locker {
    private final int maxCount;
    private int count;

    public Locker(int count) { maxCount = count; }

    public synchronized void lock() throws InterruptedException {
        while (count >= maxCount) {
            wait();
        }
        count++;
    }

    public synchronized void unlock() {
        if (count > 0) {
            count--;
            notify();
        }
    }
}

(11 P, each error gives 1 P penalty)