Nachdenkzettel: Exceptions

1. Was ist falsch mit folgendem Code? Benennen Sie die Fehler und beheben Sie diese.

```
public void doSomething() {
    try {
        lock(); // lock some resource
        // open some resource
        // try to change some things
        // fool around a bit
    } catch (Exception e) {
    } catch (ConcurrentModificationException e) {
        System.out.println("bad stuff going on today!") }
    finally {
        return;
    }
}
```

2. Was ist die Ausgabe des folgenden Programms?

```
public class TestException1 {

    public static void main(String[] args) {
        try {
            badMethod();
            System.out.print("A"); }
        catch (Exception ex) {
            System.out.print("B"); }
        finally {
            System.out.print("C");
        }

        System.out.print("D");

        }

        public static void badMethod() {
            throw new Error();
        }
}
```

3. Wann sollten Sie einen re-throw einer `Exception` implementieren?

4. In einem Web-Shop wird zur Laufzeit festgestellt, dass eine Kunden-ID nicht in der Datenbank vorhanden ist. Ist dies

- eine System-Exception?
- eine Custom-Exception?
- keine Exception?

Warum?

5. Was ist der Vorteil von Exceptions gegenüber dem Auswerten von Fehlerwerten im Return?

Nachdenkzettel

-Exceptions-

Aufgabe 1)

① Ein Catch-Block sollte nicht leer sein, es sollten zumindest über einen Print od. ähnliches Informationen über den Fehler ausgegeben werden.

② Wenn es mehrere catch-Blöcke gibt, sollte der mit der allgemeinsten Exception immer am Schluss stehen, da spezifischere Exceptions sonst nie erreicht werden.

③ Das return im finally Block kann dazu führen, dass Exceptions die aufgetreten ignoriert werden, da finally immer ausgeführt wird, auch, wenn eine Exception auftritt.

korrigierter Code:

```
public void doSomething() {
    try {
        lock(); // lock some resource
        // open some resource
        // try to change some things
        // fool around a bit
    } catch (ConcurrentModificationException e) {
        System.out.println("bad stuff going on today!");
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //some stuff for example unlocking the resource
    }
}
```

Aufgabe 2)

Die Ausgabe des Programms ist: C

Aufgabe 3)

Zum Beispiel dann wenn ich die Exception weiterreichen will um sie an anderer Stelle abzufangen, oder wenn ich eine spezifischere Exception aufrufen will, die besser auf den Fehler passt.

Aufgabe 4)

Eine System-Exception, da der Fehler bei der Abfrage der Datenbank auftritt, was über input und output streams geregelt wird.

Aufgabe 5)

Der Vorteil ist, dass Exception genaueren Aufschluss darüber geben um was für eine Art von Fehler es sich handelt. Zudem müsste man jeden Wert aus einem return überprüfen ob er gültig ist oder nicht, was sich bei Exceptions viel einfacher über den try-catch-Block realisieren lässt.