# Part 2 – Answers

## 1.

Writing tests before writing the API saves you from wasting time on writing code/specs you’ll later throw away. Continuing writing tests throughout the process of creating the API prevents surprises and provides example code that uses the API – that can be used for the documentation of the API.

## 2.

1. When writing the initial spec of the API – you should keep it as short as possible (in order to allow doing modifications quickly). This follows the key idea of spartanization – making things as short/small as possible. An API should be “as small as possible but no smaller”, and when in doubt whether a given functionality is needed – we should leave it out.
2. Minimize the accessibility of everything, e.g. make the classes and members as private as possible. Also minimize mutability - classes should be immutable, unless there’s a good reason for them not to be.
3. The API should be written such that people would be able to use it without repeatedly consulting the documentation. One specific rule to follow - functions that change the state should have a name with a verb (e.g. “clearInterruptStatus” instead of “interrupted” in the example from the lecture).

## 3.

Using Checked Exceptions is bad because it forces the user to deal with each such exception (catch/declare it). It turns out 99% of the times, the user wouldn’t want to deal with exceptions thrown by library code, and instead would prefer to simply get an immediate indication of the failure (if failure is even possible at all).

Also, having checked exceptions in an API might make implementation “leak” into the interface (in case the exception types are implementation specific).

In the lecture we are given the example of the clone function in Cloneable interface (in java), throwing a CloneNotSupportedException – which is a checked exception, and it’s a bad idea. It forces us to catch the exception each time we use the clone function, even when we know for a fact that this exception would not be thrown (because the object we’re cloning is indeed an instance of a class implementing Cloneable).