9.1 What is the difference between subtyping and subclassing? Provide an example for your explanation.

9.1.1 Subtyping

A subtype is more specific type, but behaves exactly like the more general expectation when calls are made through the interface. An example for this would be an interface car which has the method drive and the subtypes Volkswagen and Porsche. Both of those are subtypes of the interface because the drive method will behave exactly the same.

9.1.2 Subclassing

A subclass is a more specific type and behaves in ways that exceed the more general expectations when calls are made through the interface. An example for subclasse would be a base class ModeOfTransport and two subclasses Car and Plane. Obviously a car can drive and a plane can fly. Therefore both of these are more specific than the ModeOfTransport which can "only" in some ordinary way.

9.2 Using the Java class-interface hierarchy given in Figure 1, explain what is the relationship between classes and interfaces.

In Anthony J.H. Simons' JOT series from the given literature (The Theory of Classification, Part 8: Classification and Inheritance) classes refer to a polymorphic family and type is one member of this family.
Based on this definition - through the inheritance of both interfaces javax.net.ssl.X509KeyManager and javax.net.ssl.KeyManager - the subclass javax.net.ssl.X509ExtendedKeyManager creates a new family of types; Additionally both of these interfaces are substitutable because javax.net.ssl.X509KeyManager is the subtype of javax.net.ssl.KeyManager.
In regards to the java.lang.Object class the javax.net.ssl.X509ExtendedKeyManager class can be a subtype of it, if and only if thre record extension and the contravariant subtyping rules are fulfilled. Otherwise it would be a subclass of java.lang.Object.

9.3 Which forms of polymorphism are used in the Java code in Listing1? Explain each of the forms.

In the first line:

```
public class Bern<TT> {
```

a parametric polymorphism is used by implementing a generic type, so that the class can be instanciated regardless of the specific type it holds. So the type of the variable hold by the class Bern is set when an object of it is instantiated.
In the 4. line of the main method:

```
b = a;
```

We have a coercion polymorphism, because the variable of *b* which is instantiated to hold a value of type float, is handed over a value of type int.
For both System.out.println statements in lines 5 and 8 in the main method we have a overloading polymorphism, because System.out.println methods can be either used with floats, integers, string etc. and for all those cases different methods are implemented within Java.

16-124-836
Marcel Matti ZAUDER

9.4 In the Java code Listing 2, explain what concept (covariance or contravariance) exists and why.

Because we expect a Number-Array to also hold floats but with this implementation, because our Array is still a Double-Array, the following code would throw a ArrayStoreException:

```
public static void main(String[] args) {
        Double[] mh = new Double[2];
        mh[0] = 100.2;
        mh[1] = 200.2;

        Number[] nm = mh;

        nm[0] = 1000.5f;

        System.out.println(nm[0]);
}
```

Because we can create a RuntimeException even though a Float is a subtype of Number, and therefore a Float value could be a part of a Number-Array, we have a covariance concept implemented in this example.