**1)**

**1. Object**

An object is a specific instance of a class.

**2. Class**

A class is a description of an object containing

e.g. members (variables for the object), a constructor, methods and more.

Example: Class "Person" has a forename, lastname, age and methods for actions,

which the a Person in our program should be able to do.

Example:

public class Cat extends Animal {  
  
 private String name;  
  
 // Constructor  
 public Cat(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public String saySomething() {  
 return "I'm a cat, miau! My name is " + name;  
 }  
}

**3. Instantiation of object (creating an object)**

Using the "new" keyword you can create a new instance of a class, an object.

This object gets stored in the heap and the objects name references to the data.

An instance of an object could be e.g. a Person with a specific name like "Hans", age "38", eyecolour "blue" etc.

Example:

Cat myCat = new Cat("Hans");

**4. Visibility (public / private / protected)**

Public: Access to member/method from everywhere

Private: Access only from the same class

Protected: Access only from one User

Example:

private String name;

**5. Member datas / methods**

Members are variables which are a part of a specific object (class),

methods are functions of a specific object.

// Member

private String name;

// Method

public String saySomething() {  
 return "I'm a cat, miau! My name is " + name;  
}

**6. Inheritance**

A class can inherit functionality from a parents class using the keyword "extends".

Example:

public class Cat extends Animal { …

**7. Interface**

Like a class, but it only contains method signatures to make sure, that if you use this

"class" you have to implement the signatured methods. So the interface doesn't contain

the implementation of the methods.

// uses keyword “interface” instead of “class”  
public interface Moveable {  
 // contains only CONSTANTS as member  
 final int *MAXMOVE* = 20;  
  
 // only abstract Methods allowed  
 public Boolean isMoved();  
}

**8. Polymorphism**

Ability of objects to take on different forms.

Example: If you have a parent class "Animal" and subclasses "Cat" and "Dog", each of the

three classes containing the method "makeNoise()", you can override the "makeNoise()"-method

from the Animal-class when you are calling the method for a cat or a dog to get different noises.

Example:

@Override  
public String saySomething() {  
 return "I'm a dog, wuff! My name is " + name;  
}

**9. Overriding**

see 8.

Example:

@Override  
public String saySomething() {  
 return "I'm a cat, miau! My name is " + name;  
}

**10. Abstract classes**

An abstract class can't be instantiated. The class "Animal" is a typical example to make

an abstract class. This protects the code from creating an Animal-object, which makes no sense,

since you have to know which animal you want to create. So using abstract classes you can

just create an object out of a subclass of Animal, like "Cat", "Dog" etc.

Example:

public abstract class Animal {  
  
 public abstract String saySomething();  
}

**3)**

**1. What programming languages you can use for Android app development?**

Java, Kotlin and C++

**2. What is .apk file?**

Android Package, archive which contains all content of the Android app.

**3. How Android system runs apps?**

Every app runs in own Linux process and every process has it’s own virtual machine, so the code of the app is isolated from other apps.   
Each app is a different Linux user with it’s own ID assigned by the system (only system knows). But it’s possible that two apps share this ID to access each other’s files. When they have the same user ID they also run in the same process and share the same VM.

**4. Name four types of Android components. Describe each.**

Activities: Used for interaction with the user.  
Services: Component running in the background to to perform work for operations and processes.  
Broadcast Receivers: Allows the app to respond to system wide broadcast announcements.   
Content Providers: Shared set of data that you can store in the file system, web, database and more.

**5. What is manifest file and what is its purpose?**

In this file components, API level and libraries, hardware and software features are declared and user permissions are identified.

**6. What are resources? Why they are needed?**

Everything which is part of the app but source code itself (e.g. images, audio files etc.)