Questions and Exercises

These questions and exercises is an opportunity to see what you've learnt from the lecture as well as practice the new things we've been talking about. In other words, these questions and exercises are completely optional but it's recomended to do them. In the end of the document you will find the answers to the questions as well as possible solutions to the exercises, note that one can solve an exercise in different ways. There will also be some suggestions about what one could code if one want to continue with some more advanced things. These suggestions will not come with a possible solution and might include things that haven't been covered in the lecture.

Question 1
To make a minecraft mod we required three different softwares(apart from the IDE, ecplise for instance). Which three? And what are they used for?

Question 2
To specify that our class is a mod we need to specify something special. How is this done?

Question 3
When loadig things in your mod you can do so in three different methods. What should these methods be called?

Exercise 1
Write a simple mod with a config file. This mod should read four different values from three different categories. There should be one boolean value, one integer, one double and one string value. Print it all out afterwards.

Exercise 2
Write a simple mod which uses a mcmod.info file with appropriate information. Make sure that it loads properly.

Further explorations
Improve the config file from Exercise 1. Make sure it is only saved, when something has changed. Add comments to the different categories and entries to describe what they are used for. Finally add a entry that is a String array to see how that works. For more information on how do to these things you might wanna take a look in Forge's Configuration class.

Answers and solutions

Answer to Question 1

The three different things we used were Minecraft Coder Pack, Forge Mod Loader and Minecraft Forge. However, Minecraft Forge installs all three for us, so it's easy to set up. MCP allows us to decompile minecraft. This means that we can get the source code from minecraft, and also a source code with proper names to make it easier. MCP also allows us to stitch it all back together. FML allows us to load the mods properly. This means that in the end we can just save our mod as a zip file and FML makes sure its loaded properly. Forge allows us to access and do a lot of stuff without having to mess with and change the vanilla minecraft code. It also contains other helpful things, such as an easy configuration system.

Answer to Question 2

Adding this annotation just before the class makes sure its a mod. We need to specify its id, its name and its version.

```
@Mod(modid = ”myid”, name = ”some name”, version = ”2.3.1b”)
```

However, if we want it to be a multiplayer mod we want to add the following as well.

```
@NetworkMod(channels = {”steve”}, clientSideRequired = true, serverSideRequired = false, packetHandler = PacketHandler.class)
```

Here we need to specify which channels we want the client and the server to talk on as well as which sides are requried. For instance, do you need to have the mod installed to join a server with it. Lastly we also need o define the packet handler which will handle the communication between the client and the server when we want to send custom data.

Answer to Question 3

The three different methods do NOT need any specific names, what they however need are the corect parameter and the event handler annotation in fron of them. Any name that is a valid method name in java, will do. It could look something like the following.

```
@EventHandler
public void preInit(FMLPreInitializationEvent event) {

}

@EventHandler
public void load(FMLInitializationEvent event) {

}

@EventHandler
public void modsLoaded(FMLPostInitializationEvent event) {

}
```

Possible solution to Exercise1

```
//Call it from here
@EventHandler
public void preInit(FMLPreInitializationEvent event) {
        ConfigHandler.init(event.getSuggestedConfigurationFile());
        System.out.println(ConfigHandler.EXAMPLE_VALUE);
        System.out.println(ConfigHandler.ENABLE_STUFF_VALUE);
        System.out.println(ConfigHandler.SOME_TEXT_VALUE);
        System.out.println(ConfigHandler.WEIRD_NUMBER_VALUE);
}
```

```
//Most things handled here
package example.config;

import java.io.File;

import net.minecraftforge.common.Configuration;

public class ConfigHandler {


        private static final String CATEGORY_USELESS = "useless stuff";
        private static final String CATEGORY_FLAGS = "some flags";
        private static final String CATEGORY_NUMBERS = "weird numbers";

        public static int EXAMPLE_VALUE;
        private static final String EXAMPLE_NAME = "example";
        private static final int EXAMPLE_DEFAULT = 5;

        public static String SOME_TEXT_VALUE;
        private static final String SOME_TEXT_NAME = "Some text";
        private static final String SOME_TEXT_DEFAULT = "Default text";

        public static boolean ENABLE_STUFF_VALUE;
        private static final String ENABLE_STUFF_NAME = "enable this thing";
        private static final boolean ENABLE_STUFF_DEFAULT = true;

        public static double WEIRD_NUMBER_VALUE;
        private static final String WEIRD_NUMBER_NAME = "Pi";
        private static final double WEIRD_NUMBER_DEFAULT = 3.141;

        public static void init(File file) {
                Configuration config = new Configuration(file);
                config.load();

EXAMPLE_VALUE = config.get(CATEGORY_USELESS, EXAMPLE_NAME,
EXAMPLE_DEFAULT).getInt();
SOME_TEXT_VALUE = config.get(CATEGORY_USELESS, SOME_TEXT_NAME,
SOME_TEXT_DEFAULT).getString();
ENABLE_STUFF_VALUE = config.get(CATEGORY_FLAGS, ENABLE_STUFF_NAME,
ENABLE_STUFF_DEFAULT).getBoolean(ENABLE_STUFF_DEFAULT);
WEIRD_NUMBER_VALUE = config.get(CATEGORY_NUMBERS, WEIRD_NUMBER_NAME,
WEIRD_NUMBER_DEFAULT).getDouble(WEIRD_NUMBER_DEFAULT);


                config.save();
        }



}
```

Possible solution to Exercise 2
Put something like the following file in your src folder.

```
[
{
  "modid" : "StevesExample",
  "name" : "Steve's Example",
  "version" : "Lecture 1",
  "credits" : "Thanks to Steve and all the students of these courses",
  "authors": [
    "Vswe"
  ],
  "description": "An example mod as a part of Vswe's Summer courses",
  "url" : "http://courses.vswe.se/?course=3&lecture=23",
  "updateUrl" : "",
  "logoFile" : "",
  "parent" : "",
  "screenshots": [
  ]
}
]
```