1.

A software development company is changing from a traditional SDLC to Agile. Explain methodology, benefits and how the developers can adapt to this methodology.

The main difference between traditional software development and an Agile methodology is the flexibility and dynamism that the agile methodology can provide. The traditional way is to split the development life cycle into a few major phases and completing them one by one. That makes it difficult for a team to go back and change something that was previously completed, and that's why people now use an agile methodology, amongst other reasons. A main concept of agile is the incremental delivery of a project and maintaining frequent communication with the client. This allows a team to more easily adapt to changes in requirements and priorities.

2.

What is MVC and explain its architecture. How does it work and benefit projects? Find advantages and disadvantages. Include diagrams in your answer.

MVC is a type of software architecture, or a design pattern depending on who you ask. The main idea of this architecture/pattern is to split different aspects of a given software into Models, Views, and Controllers. Models manage behaviors and data. Views render model data and provide user interfaces to interact with this data. Controllers control the flow of the application, taking in user inputs and sending appropriate views, accessing the needed models when needed.

The main benefit of this in my mind is the separation of concerns. A team can easily split between user interface developers working on views and back end developers working on models or controllers. Once the concept is understood well enough, it actually becomes easier to use an MVC pattern rather than an old school mess of files and spaghetti code.

3.

What is JSP? What are the different tags in JSP? What are the benefits of using JSP in a project? Compare JSP vs HTML

JSP stands for java server page. Using a JSP page instead of a traditional HTML page allows a developer to embed java code in said page. Typically, no actual business logic or database access is written in JSP pages. This is done through servlets and models. These elements combined produce dynamic web pages. However, JSP pages are compiled to servlets before compilation, so all this could also be done in a JSP itself, however that is most likely a terrible idea. In a JSP page one can make use of "scriptlets", in which java code can be imbedded. These scriptlets contain regular java code surrounded by certains tags. These are:

- <% {code} %> For plain old java code
- <%@ {code} %> The @ symbol is for importing packages to the page
- <%= {code} %> The = sign "basically" calls the .toString() method of the object in between the tags.
- <%! {code} %> And the ! sign is to declare global variables.

Side note, these scriptlets are deprecated and the internet highly discourages using them for multiple reasons. JSTL is the most recommended alternative which should probably be taught in this course.

4.
What is Maven?
It is a build automation tool. Basically, it makes a developer's life easier. Maven can dynamically download plugins and dependencies for a project amongst other things.
I am not going to add screenshots for this tutorial.
Click file, new, other, search "maven", click Maven Project, done.
This will create an already set-up Maven project with a pom.xml file. In that file you will find information about the project, and any dependencies required.

```
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.11</version>
    </dependency>
```

Those 5 lines are all that is required for Maven to find, download, and add a dependency to your project.

5.
What are the benefits of using Spring over J2EE applications? Also, find some benefits of having Hibernate in a project.

Spring provides many things. Things that have been shown to be better and more improved then just J2EE. Dependency injection, integrated technologies like ORM frameworks, transaction management, etc.…

If a database is required in a project, which it usually is, and it is used any more than trivially, then using an object-relational mapping tool is often recommended. Old-school JDBC code is difficult to debug and prone to human error. It is also absent of features that ORM tools have, such as lazy loading and different types of optimization. However, after doing some research, I've learned that there are indeed trade offs to using something like Hibernate. Simply put, Hibernate is much more complex than what it might initially seem like, and when working on actual projects, a lot of work and a lot of experience is needed to minimize the downsides of using it.