

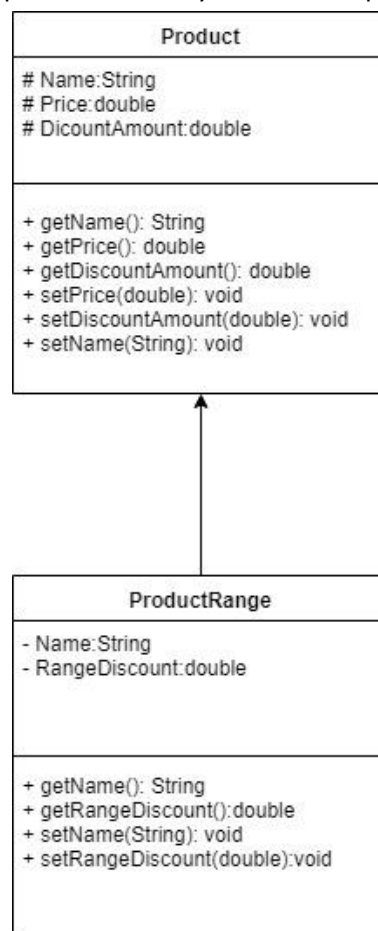
Agile Exam**Question 1**

a) The features of this product should be written in the format of user stories which can then be mapped out by the design team. The user stories for this software are as follows:

- “As an employee, I want to be able to browse the furniture products that my company produces so that I have a way of seeing all of our available products.”
- “As an employee, I want to be able view all of the ranges of any product so that I can easily view and manage each of these.”
- “As an employee, I want to be able to apply discounts to certain products if they fall under certain conditions so that I can price products accordingly with what is needed.”

These stories will then be typically placed along a user story map, which will demonstrate the flow of the system. All of the above features will be placed horizontally from left to right, showing the path of the employee user in the system. These stories combine to make the ‘minimum viable product’ (MVP) which is the minimum required in order to make the system functional for the company.

b) The software product can be broken down into an object-oriented design so that it is easy to identify all of the key components of the system and map them out accordingly.



Above is a class diagram representing the hierarchy of a product and a range of that product, which would extend the product class. Both classes have their own ability to apply discounts, either to the whole range of products, or to that one particular range. All of the getter and setter methods are public as they will need to be used to alter information. All of the class attributes are protected in the Product class and private in the ProductRange class, so that they can only be viewed and altered using the accessor methods. The class diagram represents the objects which are defined in the criteria for the company.

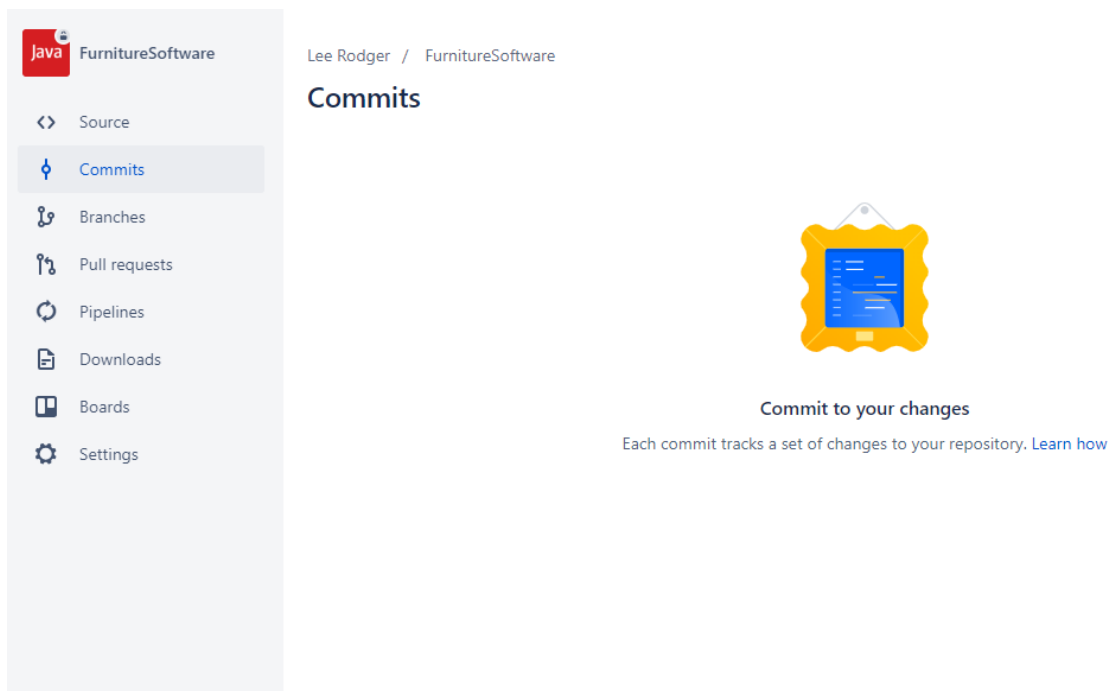
Question 2

- a) The software company can use GIT as a means of version control and accessing and updating code for the product. Firstly, they would create a remote repository so that all of the team members have access. The repository would have private access meaning that each member of the team would need to be added manually in order to view the code, and clone the repository.

The screenshot shows the 'Create a new repository' form in GitHub. The form is titled 'Create a new repository' with a link to 'Import repository' on the right. The form fields are as follows:

- Owner:** A dropdown menu showing 'LeePrototype'.
- Repository name:** A text input field containing 'FurnitureSoftware'.
- Access level:** A checkbox labeled 'This is a private repository' is checked. Below it, text says: 'Uncheck to make this repository public. Public repositories are open source and can be viewed by anyone.'
- Include a README?:** A dropdown menu showing 'No'.
- Version control system:** Two radio buttons: 'Git' (selected) and 'Mercurial'.
- Advanced settings:** A section header with a dropdown arrow.
- Description:** A large text area for the repository description.
- Forking:** A dropdown menu showing 'Allow only private forks'.
- Project management:** Two checkboxes: 'Issue tracking' and 'Wiki', both of which are unchecked.
- Language:** A dropdown menu showing 'Java'.

At the bottom right of the form are two buttons: 'Create repository' (in blue) and 'Cancel'.



As shown above, a repository is made using BitBucket. This is where all team members will push their code so that the latest version can always be downloaded by another team member. Each member can access this and clone the remote repository using the git clone command as shown below.

```
1809707@50180 MINGW64 /u/Software
$ git clone https://LeePrototype@bitbucket.org/LeePrototype/furnituresoftware.git
Cloning into 'furnituresoftware'...
```

```
1809707@50180 MINGW64 /u/Software/furnituresoftware (master)
$ |
```

- b) By using GIT, the software development team can create code within their own private repository. Once the feature has been implemented, the code can be pushed into a new branch in the public repository, where the developer can then create a pull request. This means that other members of the team can pull that code from the public repository in order to test whether or not it is ready to be merged into the main project with the rest of the working code. Once the code is tested and deemed finished, the team leader will then usually merge this code with the master branch, where all of the working code exists.

By creating a new branch, the developer is risk free of pushing the new code into the master branch, and causing problems by having non-working code.

- c) The software product should be developed within a team using an agile methodology such as SCRUM. This is ideal for products developed within teams that will be working using a public/private repository system. SCRUM is comprised of working in sprints. A new feature is defined at the start of each sprint, and a product backlog is created with all of the required features that will need to be implemented.

During a sprint, each team member can choose a user story to work on and implement. At the start of each day, the development team will conduct a 'stand-up' whereby each team member states the following:

- What they did yesterday.
- What they are going to do today.
- And any blockers which may stop them or cause an issue for them to continue.

This allows team members to determine where all members are in the development process and keeps everyone on track. Once a story is completed, this is marked on a 'burndown chart'. This chart starts with the amount of stories which are required to be developed, with the goal being to reach 0 by the end of the sprint. The length of a sprint is determined by the team. A usual time would be something such as 2-weeks.

The SCRUM methodology is ideal for iterative development where new features are being implemented and tested on a regular bases. The advantages of this methodology are that if something goes wrong, it is often a lot quicker and cheaper to make changes. The team will also be in regular communication with the product owner, so any changes required and feedback is always taken into account, allowing the team to deliver the best product possible for the client.

After each sprint, the team can also work together to determine if they planned out too much or too little, and work on adjusting the amount of work carried out in the next sprint, with the goal to always be working at the optimal pace without compromising the quality of the product.

Question 3

- a) Below is a demonstration of creating a new branch in order to create a new feature in git. The user has already cloned the public repository and has created a new branch called 'feature' in order to implement without effecting the master branch.

```
1809707@S0180 MINGW64 /u/Software/furnituresoftware (master)
$ git branch feature

1809707@S0180 MINGW64 /u/Software/furnituresoftware (master)
$ git checkout
feature          FETCH_HEAD      HEAD          master        origin/master

1809707@S0180 MINGW64 /u/Software/furnituresoftware (master)
$ git checkout feature
Switched to branch 'feature'

1809707@S0180 MINGW64 /u/Software/furnituresoftware (feature)
$ |
```

- b) Below is Java code to enable a 17% discount to a company order if the product is a repeat order.

```
public class Feature {
    public Product applyIndustrialDiscount(Product product) {
        if (repeatOrder()) {
            //17% Discount applied
            return product.getPrice() * 0.83;
        }

        //No discount applied
        return product.getPrice();
    }

    /**
     * Check if applicable for a discount
     */
    public boolean repeatOrder(Product product) {
        if (product.isReorder()) {
            return true;
        }
        return false;
    }
}
```

- c) Below is evidence of adding a new file to a branch and committing and pushing this to the remote repository.

```
1809707@S0180 MINGW64 /u/Software/furnituresoftware (feature)
$ git add Feature.java
```

```
1809707@S0180 MINGW64 /u/Software/furnituresoftware (feature)
$ git commit -m 'Commit with new feature'
```

```
1809707@S0180 MINGW64 /u/Software/furnituresoftware (feature)
$ git push --set-upstream origin feature
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 748 bytes | 374.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0)
remote:
remote: Create pull request for feature:
remote:  https://bitbucket.org/LeePrototype/furnituresoftware/pull-requests/new?source=feature&t=1
remote:
To https://bitbucket.org/LeePrototype/furnituresoftware.git
 * [new branch]      feature -> feature
Branch 'feature' set up to track remote branch 'feature' from 'origin'.
```

This is then visible on the bitbucket commit history. It can be seen that this commit was made on the feature branch.

Lee Rodger / FurnitureSoftware

Commits

 All branches ▾
 Find commits

Author	Commit	Message		Date	Builds
 Lee Rodger (18...	7e3aa8d	Commit with new feature	 feature	3 minutes ago	
 Lee Rodger (18...	0766dde	Commit with new Feature	 feature	4 minutes ago	
 Lee Rodger (18...	625bd4e	First Commit		18 minutes ago	

Question 4

- a) Another user is added to the remote repository as evidenced below.

Users

Username or email address	Write	▼	Add
 Lee Rodger	owner		
 jaymie	READ	WRITE	ADMIN 

- b) By using BitBucket, we can see the branches which have been created and the commit history, demonstrating who committed, when and a commit message.

Create branch

Filter by: **Active** Merged Find branches

Branch	Behind	Ahead	Updated	Pull request	Builds
master MAIN BRANCH			23 minutes ago		
feature		2	8 minutes ago	Create	

Lee Rodger / FurnitureSoftware

Commits

 All branches ▾
 Find commits

Author	Commit	Message		Date	Builds
 Lee Rodger (18...	7e3aa8d	Commit with new feature	 feature	3 minutes ago	
 Lee Rodger (18...	0766dde	Commit with new Feature	 feature	4 minutes ago	
 Lee Rodger (18...	625bd4e	First Commit		18 minutes ago	

- c) Compartmentalised testing is used to test sections of code. Unit testing is carried out in order to test whether or not a function is working correctly, and giving the expected output. To carry out these tests, sometimes developers need to introduce objects which may have not yet been fully implemented. For example, a fake class can be used to represent a class at the top of a hierarchy. This class contains no data, but is simply there to allow the code to execute correctly. Mock classes can also be created to test functions, which may contain limited data. For example, data which is required to pass a test can be put in. This is known as a 'double' as it contains actual data, but is manually inserted to allow the function to run successfully. These functions will eventually use the real data of the software product, but these need to be initially tested to ensure that when the other features are added, the code will work straight away without issue.

An example of unit testing using fake/mock objects and data can be seen below in Java Code.

The product class is created as a mock object, allowing us to test a function in the Feature class.

```
public class Product() {  
  
    //This method will always return true for the purposes of testing the  
    //new feature code.  
    public boolean isReorder() {  
        return true;  
    }  
}
```

The FeatureTest class is created for the purposes of unit testing a function in the feature class.

```
public class FeatureTest() {  
  
    @Test  
    public void testRepeatOrder() {  
        //Instance of the new feature class  
        Feature feature = new Feature();  
  
        //Instance of the mock object, containing one method, returning true always  
        //for the time being.  
        Product product = new Product();  
  
        //The assertion test, expecting the function repeatOrder() to return true;  
        assertEquals(true, feature.repeatOrder(product))  
    }  
}
```

Below is the Feature class as previously demonstrated, showing the function repeatOrder() which is being tested to see if it returns true, given the mock object Product.

```
public class Feature {  
    public Product applyIndustrialDiscount(Product product) {  
        if (repeatOrder()) {  
            //17% Discount applied  
            return product.getPrice() * 0.83;  
        }  
  
        //No discount applied  
        return product.getPrice();  
    }  
  
    /**  
     * Check if applicable for a discount  
     */  
    public boolean repeatOrder(Product product) {  
        if (product.isReorder()) {  
            return true;  
        }  
        return false;  
    }  
}
```