**Replacing constructor with creation method**

**Scenario**

Jack Wilson is the senior manager and owner of the BallHiring company, having over 50 employees under him. Jack has advised that their system as been running extremely slow and the sometimes-certain function and features in the system are not working as expected. After looking into the system, the software engineered identified the need for refactoring as the was an extensive amount of constructor causing confusion in the system.

**Scrum meeting discussion points**

Product demonstration to the client (check git)

Retrospective and discussion of next sprint planning

Next sprint planning (Product owner will prioritise the next two task)

Burn down chart on the number of tasks done and how many remains vs time - possible but not applicable for this assignment

Further information on the refactoring method, reasoning, risk and more

as per document on attachment.

code link: https://github.com/team-a-prt453/team-a.git

**Reasoning for refactoring/ motivation**

Having a system with too many constructors makes it difficult for end user to decipher the purpose of each contractor. This is because constructors do not communicate their intentions efficiently and effectively. If the system as too many constructors, it is impossible to add a new constructor with the same arguments. Present dead and unused constructors in the system affects the system and causes baffling.

**Solution**

The way to resolve this by replacing each of the constructors with intentional creation method which will return object instances. These creation methods can be static or non-static and the creation method can also be moved to a factory object.

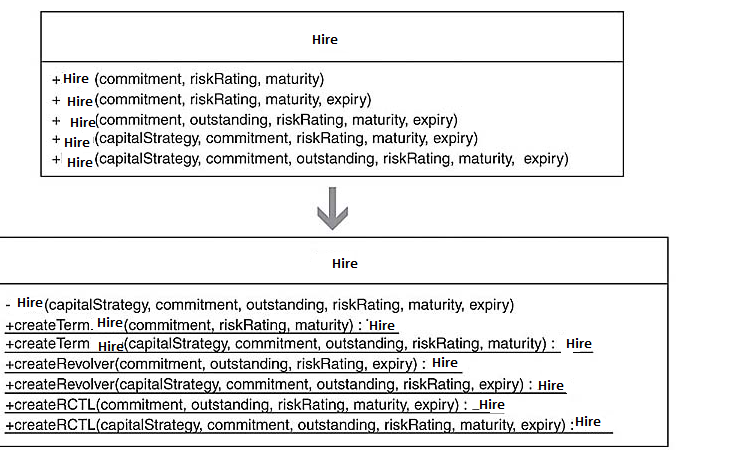
**Advantages**

* There are no name constraints on Creation Methods, so you can name them to clearly express what you are creating (e.g., createTermHire() or createRevolver()). This naming flexibility means that two differently named Creation Methods can accept the same number and type of arguments.
* The code is now more reliable
* Allows easy add of new creation method
* If there are too many creation methods, it is now easy to move all to factory classes.

**Disadvantages**

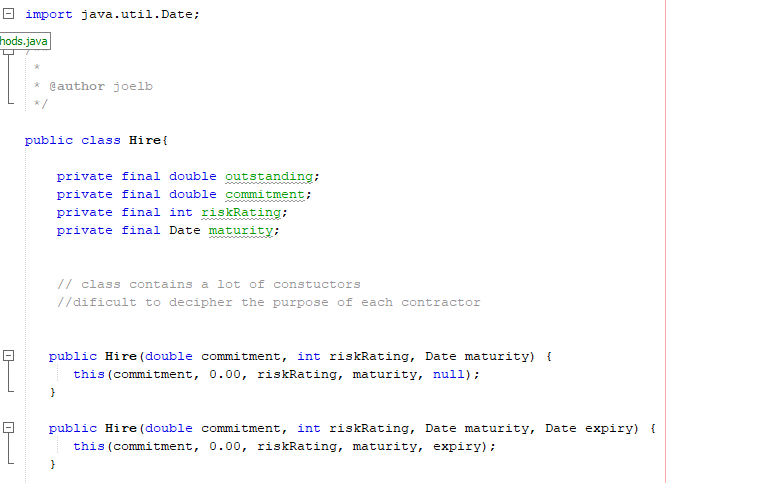
* This makes the creation non-static

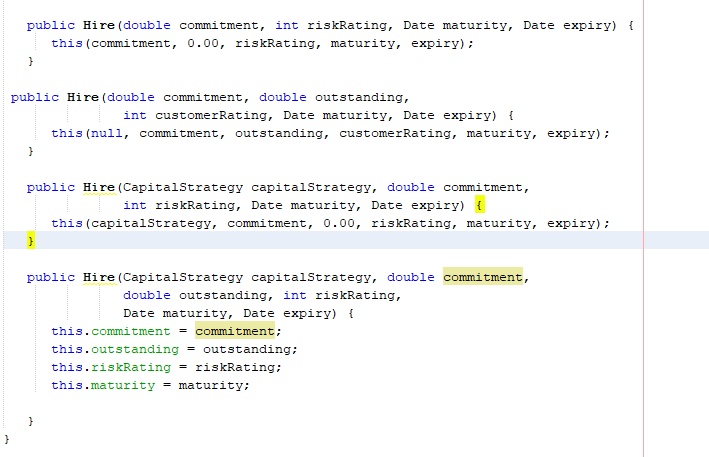
**UML diagram**



**Current code**

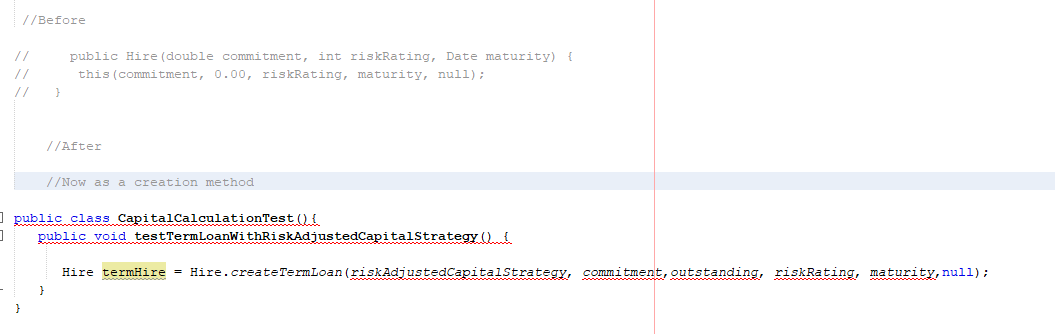
The hiring class contains way to many constructors as a result, we cannot differ the intentions and purpose of each constructor.

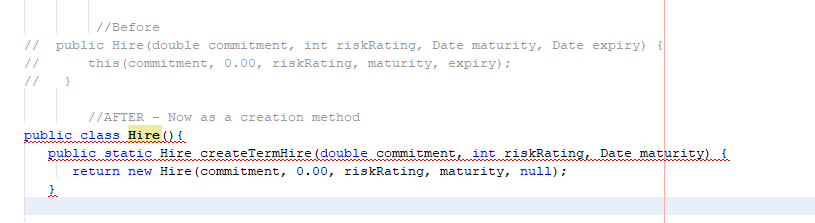




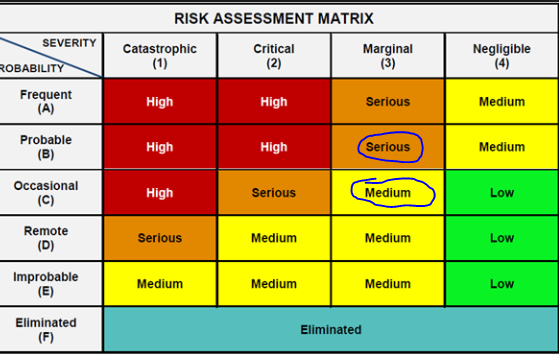
**After**

They are multiple different hiring services which all play different task therefore, first identify a class constructor in order to create an instance of it. then use the extract method to make it a public and static method. Now the move method can be applied on the creation method.





This process is repeated for all constructor until they are all turned into creation methods. If a constructor on the class has no callers outside the class, make it non-public as it does not need to be accessed from the outside.

**Risk**

The risk of having the over load constructor’s problem in a hiring system is low however, with 50 employee this system in in charge of all the work that is done by the business therefore a high priority.

**Testing phase**

As part of the testing process of the software can be testing after each iteration to ensure the it is always ready for deployment. The system needs to remain online even during the changes.

**Tdd** – testing process can be applied and continuous integration

