**Difference between update and merge:**

* The update() method works with persistent objects only, while the merge() method can work with both persistent and detached objects.
* The update() method updates the object's state in the database and returns void, while the merge() method merges the state of the object

**First level And Second level cache:**

the first-level cache is a mandatory cache that is created for each Session, and it stores objects retrieved or saved during that Session. The second-level cache is an optional cache that can be shared across multiple Sessions, and it provides a global cache for storing frequently accessed data.

**Difference between get and load:**

The "get" method is used to fetch an object from the database by its primary key. If the object is not found, the "get" method will return null.  
If the object is not found, the "load" method will throw an ObjectNotFoundException.

The "load" method, on the other hand, is used to lazily fetch an object from the database by its primary key.

**Difference between save and persist**

"save" is more flexible and can be used for both transient and persistent instances, while "persist" is specifically designed for making transient instances persistent.

The "save" method is used to save an object to the database and return its unique identifier. If the object already exists in the database, it will be updated with the new data.

The "persist" method, on the other hand, is used to make a transient instance persistent.

**Cohesion**

Cohesion is a measure of the degree to which the elements within a single class are related to each other

Copuling:

Coupling in Java refers to the degree of interdependence between software modules or classes

1. Data coupling: Occurs when classes share data through parameters or return values. It is the least restrictive form of coupling and can often be found in well-designed object-oriented systems.

Reducing coupling between classes is an important aspect of software design because it can lead to increased flexibility, reusability, and maintainability.

**Why Wait, Notify, and NotifyAll are in Object Class.**

Wait and notify is not just normal methods or synchronization utility, more than that they are **communication mechanism between two threads in Java**. And Object class is the correct place to make them available for every object if this mechanism is not available via any java keyword like synchronized.

Synchronized is to provide mutual exclusion and ensuring [thread safety of Java class](http://javarevisited.blogspot.com/2012/01/how-to-write-thread-safe-code-in-java.html) like race condition while wait and notify are communication mechanism between two thread.

**Locks are made available on per Object basis**, which is another reason wait and notify is declared in Object class rather then Thread class.

Thread vs runnable:

[Java doesn't support multiple inheritances](http://javarevisited.blogspot.com/2011/07/why-multiple-inheritances-are-not.html), which means you can only extend one class in Java so once you extended the Thread class you lost your chance and can not extend or inherit another [class in Java](http://javarevisited.blogspot.com/2011/10/class-in-java-programming-general.html)

**Runnable**means we can reuse the task and also has the liberty to execute it from different means. since you can not restart a Thread once it completes.

Aggregation vs composition

If A and B two classes are related to each other such that, B ceased to exist when A is destroyed, then the association between two objects is known as **Composition**. An example is Car and Engine. While if A and B are associated with each other, such that B can exist without being associated with A, then this association is known as **Aggregation**.

Association  A---->B  
Composition  A-----<filled>B  
Aggregation  A-----<>B

Association is a semantically weak relationship (a semantic dependency) between otherwise unrelated objects. An association is a “using” relationship between two or more objects in which the objects have their own lifetime and there is no owner.

As an example, imagine the relationship between a doctor and a patient. A doctor can be associated with multiple patients. At the same time, one patient can visit multiple doctors for treatment or consultation. Each of these objects has its own life cycle and there is no “owner” or parent

**What is the difference between FileInputStream and FileReader in Java?** ([detailed answer](http://javarevisited.blogspot.sg/2014/04/difference-between-fileinputstream-and-filereader-in-java.html))  
The main difference between FileInputStream and FileReader is that the former is used to read binary data while the latter is used to read text data,

Throw vs Throws:

) You can declare multiple exceptions thrown by the method in throws keyword by separating them in common e.g. throws IOException, ArrayIndexBoundException, etc, while you can only throw one instance of exception using throw keyword

Diamond Problem in java:

 First reason is **ambiguity around the Diamond problem**, consider a class A has foo() method and then B and C derived from A and has their own foo() implementation, and now class D derives from B and C using multiple [inheritance](http://javarevisited.blogspot.com/2012/10/what-is-inheritance-in-java-and-oops-programming.html) and if we refer just foo() compiler will not be able to decide which foo() it should invoke.   
  
This is also called the Diamond problem because the structure on this inheritance scenario is similar to 4 edge diamond, see below

           A foo()

           / \

          /   \

   foo() B     C foo()

          \   /

           \ /

            D

           foo()