**Q1) What is an immutable class?**

**Ans)** Immutable class is a class which once created, it’s contents can not be changed. Immutable objects are the objects whose state can not be changed once constructed. e.g. String class

**Q2) How to create an immutable class?**

**Ans)** To create an immutable class following steps should be followed:

1. Create a final class.
2. Set the values of properties using constructor only.
3. Make the properties of the class final and private
4. Do not provide any setters for these properties.
5. If the instance fields include references to mutable objects, don't allow those objects to be changed:
   1. Don't provide methods that modify the mutable objects.
   2. Don't share references to the mutable objects. Never store references to external, mutable objects passed to the constructor; if necessary, create copies, and store references to the copies. Similarly, create copies of your internal mutable objects when necessary to avoid returning the originals in your methods.

public final class FinalPersonClass {

private final String name;

private final int age;

public FinalPersonClass(final String name, final int age) {

this.name = name;

this.age = age;

}

public int getAge() {

return age;

}

public String getName() {

return name;

}

}

**Q3) Immutable objects are automatically thread-safe –true/false?**

**Ans)** True. Since the state of the immutable objects can not be changed once they are created they are automatically synchronized/thread-safe.

**Q4) Which classes in java are immutable?**

**Ans)** All wrapper classes in java.lang are immutable –   
String, Integer, Boolean, Character, Byte, Short, Long, Float, Double, BigDecimal, BigInteger

**Q5) What are the advantages of immutability?**

Ans)

* Immutable objects are automatically thread-safe, the overhead caused due to use of synchronisation is avoided.
* Once created the state of the immutable object can not be changed so there is no possibility of them getting into an inconsistent state.
* The references to the immutable objects can be easily shared or cached without having to copy or clone them as there state can not be changed ever after construction.
* The best use of the immutable objects is as the keys of a map.