**Immutable Class**

An object is immutable if its state cannot change after construction. Immutable objects don’t expose any way for other objects to modify their state; the object’s fields are initialized only once inside the constructor and never change again.

With immutable classes, states are never modified; every modification of a state results in a new instance, hence each thread would use a different instance and developers wouldn’t worry about concurrent modifications.

**How Do We Create an Immutable Class**

In order to create an immutable class, you should follow the below steps:

1. Make your class ***final,***so that no other classes can extend it.
2. Make all your fields ***final,***so that they’re initialized only once inside the constructor and never modified afterward.
3. Don’t expose setter methods.
4. When exposing methods which modify the state of the class, you must always return a new instance of the class.
5. If the class holds a mutable object:
   * Inside the constructor, make sure to use a clone copy of the passed argument and never set your mutable field to the real instance passed through constructor, this is to prevent the clients who pass the object from modifying it afterwards.
   * Make sure to always return a clone copy of the field and never return the real object instance.

Example:

public final class ImmutableStudent {

private final int id;

private final String name;

public ImmutableStudent(int id, String name) {

this.name = name;

this.id = id;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

}

### Passing Mutable Objects to Immutable Class

public class Age {

private int day;

private int month;

private int year;

public int getDay() {

return day;

}

public void setDay(int day) {

this.day = day;

}

public int getMonth() {

return month;

}

public void setMonth(int month) {

this.month = month;

}

public int getYear() {

return year;

}

public void setYear(int year) {

this.year = year;

}

}

public final class ImmutableStudent {

private final int id;

private final String name;

private final Age age;

public ImmutableStudent(int id, String name, Age age) {

this.name = name;

this.id = id;

this.age = age;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public Age getAge() {

return age;

}

}

public static void main(String[] args) {

Age age = new Age();

age.setDay(1);

age.setMonth(1);

age.setYear(1992);

ImmutableStudent student = new ImmutableStudent(1, "Alex", age);

System.out.println("Alex age year before modification = " + student.getAge().getYear());

age.setYear(1993);

System.out.println("Alex age year after modification = " + student.getAge().getYear());

}

Test 2:

public ImmutableStudent(int id, String name, Age age) {

this.name = name;

this.id = id;

Age cloneAge = new Age();

cloneAge.setDay(age.getDay());

cloneAge.setMonth(age.getMonth());

cloneAge.setYear(age.getYear());

this.age = cloneAge;

}