# Lab 3 – Mutability

“Immutable objects are simple. They can only be in one state, which is carefully controlled by the constructor. One of the most difficult elements of program design is reasoning about the possible states of complex objects. Reasoning about the state of immutable objects, on the other hand, is trivial.   
  
Immutable objects are also safer. Passing a mutable object to untrusted code, or otherwise publishing it where untrusted code could find it, is dangerous — the untrusted code might modify its state, or, worse, retain a reference to it and modify its state later from another thread. On the other hand, immutable objects cannot be subverted in this manner by malicious or buggy code, so they are safe to share and publish freely without the need to make defensive copies.”

—**Brian Goetz,** Java Concurrency in Practice

## Introduction

An immutable object is one whose externally visible state cannot change after it is instantiated. The String, Integer, and BigDecimal classes in the Java class library are examples of immutable objects -- they represent a single value that cannot change over the lifetime of the object.

## Lab Instructions:

**Consider the following code:**

**import** java.util.Date;

**public** **final** **class** Course {

String courseName;

**private** **final** Date startDate;

**public** Course (String courseName, Date startDate) {

**this**. courseName = courseName;

**this**. startDate = **new** Date(startDate.getTime());

}

**public** String getCourseName() {

**return** courseName;

}

**public** Date getStartDate() {

**return** **new** Date(startDate.getTime());

}

}

1. Is this program mutable?
   * If you think it is, what changes would need to be implemented (to make it immutable) and why?
   * If you think it is already immutable, explain why?
   * Show your code – Create a **separate** tester class, create a new course and see if you can manipulate the course details.
2. Did you notice anything about the Date object? Do some research and explain your findings.
3. Remember to write your **conclusions** on your work.

## Submission and Grading Details

This lab will form part of your complete Lab book, which must be submitted at the end of the module. You must reference your work. You should follow the layout of a typical lab book, and add extra headings as necessary. **Sample** layout below:

* Lab 3 Mutability
  + Description
  + Aims
  + Method
  + Results
    - Immutable Code or Mutable Code?
    - Date Object
  + Conclusions
  + Appendix
  + References

|  |  |
| --- | --- |
|  | **Marks Available** |
| Description | 5 |
| Aims | 5 |
| Method | 10 |
| Results/Testing/Evaluation |  |
| * Is code Mutability or Not, explain. * Tester Class Code | 25  10 |
| * Research Date Object | 20 |
| Conclusion | 25 |
|  |  |
| **TOTAL** | **100** |