**CS2030 Important Notes**

1. **The 4 OOP Principles**

* Abstraction
* Encapsulation
* Inheritance
* Polymorphism

1. **What happens ONLY during compilation time?**

* **Type inference** - inferring the type of a variable whose type is not specified. (in generics)
  + e.g. ArrayList<Integer> list = new ArrayList<>();
  + <https://www.javatpoint.com/java-8-type-inference>
* **Type erasure** - replacing a type parameter of generics with either Object or its bound

1. **What happens ONLY during runtime?**

* **Late binding** - determine which instance method to call depending on the type of a reference object

1. What happens at compile time **and** runtime?

* **Type checking** - checking if the value matches the type of the variable it is assigned to
  + Why does type checking also occur during runtime even though it is a static typing language?
  + The reason is due to reasons sth like the following code:
  + public static void main(String[] args){  
     Object[] array = Math.random()<.5? new String[2]: new Object[2];  
     array[0] = "Hello, World!";//compiler knows this is safe  
     System.out.println(array[0]);  
     array[1] = new Object();//compiler must check array type  
    }
  + <https://stackoverflow.com/questions/43774681/java-dynamically-checking-type-of-an-object>
  + <https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html>
  + In the code above, there is no way there we can know the exact type of the array until we run the code, hence the line array[1]] = new Object() may be unsafe (unsafe as in storing Object in a String array, which you shouldn't do so). Hence, additional type checking also need to be done during runtime to deal with this kind of issue.

* **Type casting - converting the type of one variable to another**
  + Type casting is **done** at compile time, but is **checked** at run time
  + e.g. the following code will pass the compiler, but will result at classcast exception during runtime

public static void main(String[] args) {

  3         Object a = Integer.valueOf(5);

  4         String b = (String) a;

  5     }

* <https://docs.oracle.com/javase/specs/jls/se7/html/jls-5.html> (Under Reference Type Casting)
* Ok, so basically I think type casting basically mean to change the type of one variable from one to another, there are some rules that applied in type casting. For the above code, during compile time it changes the type of a which is Object to String so that the reference a can be assigned to b. This is allowed at compile time as an Object has the possibility of being converted into String. However, whether this will succeed or not depends on whether the object that a is pointing to is truly a string. Therefore, extra check is needed during runtime to see if a is really a string so that it can be assigned to b.

<https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html>

* **Accessibility checking - checking if a class has an access to a field in another class**
  + As we know, accessibility checking is usually done during compile time. Let's say you try to access a private static attribute of class A in class B, you wouldn't able to compile class A.
  + **Why it also happens at runtime?**
  + Here's the explanation. But first, let me introduce you sth call IllegalAccessError, which happens when a instance trying to access the attributes of another class that are not allowed to access
  + Basically this is done during runtime as well is to deal with the case where **not all classes are compiled**.
  + Suppose we have two classes, class A and class B. At first, I set an attribute called public String *message* in A and I try to access it in B. I then compiled both classes and this wouldn't give rise to any error since *message* is public. However, if I change the attribute to **private** in class A and I **only** compile class A, class A will be compiled fine, but now class B has problem (since I can't access *message* of class A in class B now), but since I don't compile class B, nothing will happen, the error still remain there. Therefore, accessibility also need to be checked during runtime to deal with this kind of error.
  + When you try to run the program, it will throw you an IlegalAccessError, which is extended from IncompatibleClassChangeError
  + <https://examples.javacodegeeks.com/java-basics/exceptions/java-lang-illegalaccesserror-how-to-resolve-illegal-access-error/>
  + <https://stackoverflow.com/questions/14077450/is-access-checked-at-compile-time-or-run-time>

<https://coderanch.com/t/368559/java/Compile-time-Type-Checking-Run>