**Generic Classes in Java**

**Overview.** Give an introduction to generic classes in Java, including their benefits, formal naming conventions for “formal types”, and how to incorporate them into code. As an example, we’ll code up a LinkedList hardcoded to store integer data in class. Then, we’ll make the appropriate changes to transform the LinkedList into a generic class. Finally, make changes to the generic LinkedList class to allow for sorted insertion into the list by requiring the the “actual type” for the LinkedList class implements Comparable<T>.

**Definitions.**

* Generic class / interface: a class / interface whose declarations have one or more “formal type” parameters.
* A generic class / interface is defined as followed: the class / interface name followed by an angle-bracketed list of “formal type” parameters (instantiated by “actual type”, e.g. List<E> is a List (raw type) of formal type E, possibly instantiated as a List of actual type String List<String>)
* You can instantiate generic types without actual types, but don't benefit from compile-time enforcement that all insertions into a collection are of the valid actual type. This was done to allow backwards compatibility with Java 1.4 and below.
* Formal Type Naming Conventions:
  + E - Element
  + K - Key
  + N - Number
  + T - Type
  + V - Value
  + S,U,V, etc. - 2nd, 3rd, 4th types.

**Why Use Generics Classes (i.e. benefits)?**

* Parameterize classes and interfaces
* Code re-use
* Stronger type checks (i.e. catch more errors at compile-time)
* Fewer casts (e.g. retrieving elements from a raw type without actual type specified)’

**Caveats:**

* Actual types cannot be primitives (e.g. int, char).
* All primitives have object types (e.g. Integer, Character).

**References:**

1. <https://docs.oracle.com/javase/tutorial/java/generics/why.html>
2. Effective Java (2nd Edition) by Joshua Bloche: <http://www.eecs.qmul.ac.uk/~mmh/APD/bloch/generics.pdf>