**Assignment 9**

**1. What is Generic Class?**

**Java Generic** methods and **generic classes** enable programmers to specify, with a single method declaration, a set of related methods, or with a single **class** declaration, a set of related **types**, respectively. **Generics** also provide compile-time type safety that allows programmers to catch invalid **types** at compile time.

You can write a single generic method declaration that can be called with arguments of different types. Based on the types of the arguments passed to the generic method, the compiler handles each method call appropriately. Following are the rules to define Generic Methods −

* All generic method declarations have a type parameter section delimited by angle brackets (< and >) that precedes the method's return type (< E > in the next example).
* Each type parameter section contains one or more type parameters separated by commas. A type parameter, also known as a type variable, is an identifier that specifies a generic type name.
* The type parameters can be used to declare the return type and act as placeholders for the types of the arguments passed to the generic method, which are known as actual type arguments.
* A generic method's body is declared like that of any other method. Note that type parameters can represent only reference types, not primitive types (like int, double and char).

**Generic Classes**

* A generic class declaration looks like a non-generic class declaration, except that the class name is followed by a type parameter section.
* As with generic methods, the type parameter section of a generic class can have one or more type parameters separated by commas. These classes are known as parameterized classes or parameterized types because they accept one or more parameters.

**Example**

public class Box<T> {

private T t;

public void add(T t) {

this.t = t;

}

public T get() {

return t;

}

public static void main(String[] args) {

Box<Integer> integerBox = new Box<Integer>();

Box<String> stringBox = new Box<String>();

integerBox.add(new Integer(10));

stringBox.add(new String("Hello World"));

System.out.printf("Integer Value :%d\n\n", integerBox.get());

System.out.printf("String Value :%s\n", stringBox.get());

}

}

**2. Benefits of Generic Class**

* Stronger type checks at compile time. A Java compiler applies strong type checking to generic code and issues errors if the code violates type safety. ...
* Elimination of casts. ...
* Enabling programmers to implement generic algorithms.
* Type safety and usability.

**3. Uses of generic class in real life scenario**

I often create business entity objects, such as a TV, and then I use List<TV>to have a list of TV... such as the sales for that day. There you are using the generic List<of Type T>.

**4. What is Hash Set?**

Java Hash Set class is used to create a collection that uses a hash table for storage. It inherits the Abstract Set class and implements Set interface.

Hash table stores information by using a mechanism called **hashing**. In hashing, the informational content of a key is used to determine a unique value, called its hash code.

The hash code is then used as the index at which the data associated with the key is stored. The transformation of the key into its hash code is performed automatically.

**5. Characteristics of hash set**

* Hash set stores the elements by using a mechanism called **hashing.**
* Hash Set contains unique elements only.
* Hash Set doesn’t maintain any order, the elements would be returned in any random order.
* Hash Set doesn’t allow duplicates. If you try to add a duplicate element in Hash Set, the old value would be overwritten.
* Hash Set allows null values however if you insert more than one nulls it would still return only one null value.
* Hash Set is non-synchronized.
* The iterator returned by this class is fail-fast which means iterator would throw ConcurrentModificationException if Hash Set has been modified after creation of iterator, by any means except iterator’s own remove method.

**6. Difference between list and set**

A **Set** cannot contain duplicate elements while a **List** can. A **List** (in Java) also implies order. Conceptually we usually refer to an unordered grouping that allows duplicates as a Bag and doesn't allow duplicates are a **Set**. **List** is used to collection of elements with duplicates.

Set uses equals () method to check uniqueness of elements stored in Set, while Sorted Set uses [compareTo() method](http://javarevisited.blogspot.com/2011/11/how-to-override-compareto-method-in.html) to implement natural sorting order of elements. In order for an element to behave properly in Set and Sorted Set, equals and compareTo must be consistent to each other.

Popular implementation of List interface in Java includes Array List, Vector and Linked List. While popular implementation of Set interface includes Hash Set, Tree Set and Linked Hash Set.

**7. Advantages of Hash Set**

The only catch of Hash Set is that there is no access by indices. To access elements you can either use an enumerator or use the built-in function to convert the Hash Set into a List and iterate through that.

It is still much faster than a regular List. Some basic benchmarks showed that Hash Set is decently faster when dealing with primary types (int, double, bool, etc.). It is a lot faster when working class objects. So that point is that Hash Set is fast.

**8. Applications of Hash Set in Real Life**

**Hash Set of Student records** where each Student record contains three fields – **name**, **roll No** and **department**. In these, roll No will be unique for all students.

It can be used for hospital application with patient’s record like patients id, patient’s health description and last visited where patient id is different for each patient.