GUAVA - OVERVIEW

Guava is an open source, Java-based library and contains many core libraries of Google, which are being used in many of their projects. It facilitates the best coding practices and helps reduce coding errors. It provides utility methods for collections, caching, primitives support, concurrency, common annotations, string processing, I/O, and validations.

Benefits of Guava

- **Standardized** - The Guava library is managed by Google.
- **Efficient** - It is a reliable, fast and efficient extension to the java standard library.
- **Optimized** - The library is highly optimized.
- **Functional Programming** - It adds functional processing capability to Java.
- **Utilities** - It provides many utility classes which are regularly required in programming application development.
- **Validation** - It provides a standard failsafe validation mechanism.
- **Best Practices** - It emphasizes on best practices.

Consider the following code snippet.

```
public class GuavaTester {
   public static void main(String args[]) {
      GuavaTester guavaTester = new GuavaTester();

      Integer a =  null;
      Integer b =  new Integer(10);

      System.out.println(guavaTester.sum(a,b));
   }

   public Integer sum(Integer a, Integer b){
      return a + b;
   }
}
```

Run the program to get the following result.

```
Exception in thread "main" java.lang.NullPointerException
 at GuavaTester.sum(GuavaTester.java:13)
 at GuavaTester.main(GuavaTester.java:9)
```

Following are the problems with the code.

- sum is not taking care of any of the parameters to be passed as null.
- caller function is also not worried about passing a null to the sum method accidently.
- When the program runs, NullPointerException occurs.

In order to avoid the above problems, null check is to be made in each and every place where such problems are present.

Let's see the use of Optional, a Guava provided Utility class to solve the above problems in a standardized way.

```
import com.google.common.base.Optional;
```

```
public class GuavaTester {
   public static void main(String args[]){
      GuavaTester guavaTester = new GuavaTester();

      Integer invalidInput = null;

      Optional<Integer> a =  Optional.of(invalidInput);
      Optional<Integer> b =  Optional.of(new Integer(10));

      System.out.println(guavaTester.sum(a,b));
   }

   public Integer sum(Optional<Integer> a, Optional<Integer> b){
      return a.get() + b.get();
   }
}
```

Run the program to get the following result.

```
Exception in thread "main" java.lang.NullPointerException
 at com.google.common.base.Preconditions.checkNotNull(Preconditions.java:210)
 at com.google.common.base.Optional.of(Optional.java:85)
 at GuavaTester.main(GuavaTester.java:8)
```

Let's understand the important concepts of the above program.

- **Optional** - A utility class, to make the code use the null properly.
- **Optional.of** - It returns the instance of Optional class to be used as a parameter. It checks passed, not to be 'null'.
- **Optional.get** - It gets the value of the input stored in the Optional class.

Using the Optional class, you can check whether caller method is passing a proper parameter or not