JAVA FUNDAMENTALS ASSIGNMENT

Q1. (a) Singleton Class

Sol:- A singleton class is a class which limits the creation of objects to one i.e, a class having only one object at a time.

Even if we try to instantiate the class more than once, the new variable will point to the first instance created. So, any changes made to the variable inside class through any instance affects the variables of single object and the changes are reflected.

Example of singleton class is as follow:-

```
public class Singleton{
        private static Singleton single_obj=null;
        private String str ;
        private Singleton(){ s=" This is a singleton class "; }
        private static Singleton getInstance()
        {
                if(single_obj==null)
                        single_obj=new Singleton();
                return single_obj;
        }
}

class Main{
        public static void main(String args[])
        {
                Singleton obj1 = Singleton.getInstance();
                Singleton obj2 = Singleton.getInstance();
                obj1.str=(obj1.str)+" object";
                System.out.println("String from obj1 is "+obj1.str);
                System.out.println("String from obj2 is  "+obj2.str);
        }
}
```

The above program will print :
String from obj1 is This is singleton class object
String from obj2 is This is singleton class object

Because the changes made by obj1 will also be reflected back to obj2 and obj2 will also print the same string.

(b) Thread-safe / Synchronization

Sol:- As we can see in above code, this code will create multiple instances of Singleton class if called by more than one thread in parallell. So to avoid this case and make the class thread-safe a process known as synchronization comes into the part.

Synchronization is the capability to control the access of multiple threads to any shared resource. We can achieve synchronization by adding keyword "synchronized" before the function name.

So, in above case synchronization can be achieved by replacing the getInstance() method by following code:

```
private static  synchronized Singleton getInstance()
{
        if(single_obj==null)
                single_obj=new Singleton();
        return single_obj;
}
```

(c) Double check locking in singleton class

Sol:- Double checked locking is a way to ensure that only one instance of Singleton class will be created through an application life cycle. For an existing object of singleton class the code is checked twice with and without locking that's why it is known as double check locking.

In double check there will be checks for first single instance, one without locking and other with locking. We can take the example of simple single class in 1(a). We can replace the getInstance() method as follows:

```
private static Singleton getInstance()
{
        if(single_obj==null){                                   // Single checked
                synchronized (Singleton.class){
                        if(single_obj ==null) {                 // Double checked
                                single_obj=new Singleton();
                        }
                }
        }
        return single_obj;
}
```

Q2. Count occurences of distinct strings in a arraylist.

Sol:-

```
import java.util.*;

public class Main
{
        public static void main(String[] args) {

                List<String> listofstrings = Arrays . asList("b", "a", "a", "a", "b", "c", "d", "c" );

                Map<String, Integer> frequency = new HashMap<>();
                for (String str: listofstrings) {
                        Integer count = frequency.get(str);
                        if (count == null)
                                count = 0;

                        frequency.put(str, count + 1);
                }

                for (Map.Entry<String, Integer> i : frequency.entrySet()) {
                        System.out.println(i.getKey() + ": " + i.getValue());
                }
        }
}
```

Output:-

Q3. NoClassDefFound Exception

Sol:- When a particular class is not found at runtime, the application throws either a ClassNotFound Exception or a NoClassDefFound Error.

ClassNotFound Exception is a run time exception that occurs when an application tries to load a class at runtime using the Class.forName() or loadClass() methods and the class with specified name is not found in the class path.

NoClassDefFound Error is an error that is thrown when the JRE tries to load the definition of class and that class definition is non longer available i.e, the definition was present at compile time but is missing at runtime..