***API’s And ANNOTATION ASSIGNMENT***

**Q1. Program to display current time and date in java?**

**Ans.** package API;  
  
import java.time.LocalDate;  
import java.time.LocalTime;  
  
public class DateTime {  
 public static void main(String[] args) {  
 LocalDate date = LocalDate.*now*();  
 System.*out*.println(date);  
 int day = date.getDayOfMonth();  
 int month = date.getMonthValue();  
 int year = date.getYear();  
 System.*out*.println(day + "/" + month + "/" + year);  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
  
 LocalTime tm = LocalTime.*now*();  
 System.*out*.println(tm);  
 int hour = tm.getHour();  
 int min = tm.getMinute();  
 int sec = tm.getSecond();  
 int nano = tm.getNano();  
 System.*out*.println();  
 }  
}

**Q2. Write a program to convert a date to string in the format “MM/dd/yyyy” ?**

Ans. package API;  
  
import java.time.LocalDate;  
  
public class DatePractice {  
 public static void main(String[] args) {  
 LocalDate dt = LocalDate.*now*();  
 System.*out*.println(dt);  
 int day = dt.getDayOfMonth();  
 int month = dt.getDayOfMonth();  
 int year = dt.getYear();  
 String dtValue = String.*valueOf*(day + "/" + month + "/" + year);  
 System.*out*.println(dtValue);  
 }  
}

**Q3. What is the difference between collections ad streams? Explain with an example.**

**Ans.** Collection and stream both can work together but they both are the different concepts, as we know that collection is basically a “collections of data” and if we do any operations on that so it will reflect on the original one also but on the other hand if we make an stream of any data and we want do any operations on that so changes will be occur on that particular stream, original data will not affect.

Here below the one snippet I added in which we can see that we have one collection in which we save the data , we can see that we also made an stream also so if we do any changes in stream original collection will not affect.

But the major difference between both of them are in stream we can do only one operation at a given time for other we have to make an another stream but in collection we can do several operations.

package API;  
  
import java.util.ArrayList;  
import java.util.stream.\*;  
public class StreamPractice {  
 public static void main(String[] args) {  
// List<Integer> list = Arrays.asList(8,2,4,7,5);  
// Stream<Integer> streamData = list.stream();  
// streamData.forEach(n -> System.out.println(n));  
 ArrayList <Integer> al1 = new ArrayList<>();  
 al1.add(8);  
 al1.add(2);  
 al1.add(4);  
 al1.add(7);  
 al1.add(5);  
 System.*out*.println(" Original collection data " + al1);  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 Stream <Integer> streamData = al1.stream();  
 // Stream<Integer> sortedData=streamData.sorted();  
 // streamData.forEach(n -> System.out.println(n));  
 // sortedData.forEach(n -> System.out.println(n));  
 //Stream <Integer> filData=sortedData.filter(n-> n%2==0);  
 //filData.forEach(n-> System.out.println(n));  
 // Stream<Integer> map =sortedData.map(n-> n\*2);  
 // map.forEach(n -> System.out.println(n));  
 Stream<Integer> finalStream = streamData.sorted().filter(n-> n%2==0).map(n-> n\*2);  
 finalStream.forEach(n-> System.*out*.println(n));  
  
 }  
}

**Q4. What is enums in java? Explain with an example.**

**Ans.** when we have a sets of value and we want to make them constant, at that time enum concepts comes into scope, the same thing we can do with final keyword, but by using final keyword only one variable can be constant but by enum by sets of value become constant.

package API;  
  
enum Result{  
 *PASS*, *FAIL*, *NR*;  
 int marks;  
 Result()  
 {  
 System.*out*.println("Constructor is in block");  
 }  
 void setMarks(int marks)  
 {  
 this.marks = marks;  
 }  
  
 int getMarks() {  
 return marks;  
 }  
}  
public class EnumPractice {  
 public static void main(String[] args) {  
 Result.*PASS*.setMarks(50);  
 int m1 = Result.*PASS*.getMarks();  
 System.*out*.println(m1);  
  
 int m2 = Result.*FAIL*.getMarks();  
 System.*out*.println(m2);  
  
 Result.*NR*.setMarks(45);  
 int m3 =Result.*NR*.getMarks();  
 System.*out*.println(m3);  
  
 }  
}

**Q5. What are inbuilt annotations in java?**

**Ans**. For General Purpose:-

* @override
* @Deprecated
* SafeVar Args
* Suppress Warnings
* Functional Interface

Meta Annotations:-

* @Inherited
* @Documented
* @Target
* @Retention
* @Repeatable