Practical-2
Running Java code on android studio and Demonstrating ActivityLifecycle

1. **Write down steps to run java program in android studio**

• From the project folder select java folder
• Right click on java folder and select new then java class.
• Provide a class name and press ok.
• Right the java code.
• Right click on java file and select run “Print.main()” with coverage.

2. **Write a java program to print 1 to 10 using for loop.**

- **Code:-**

```
package com.example.lab_2;

public class forloopdemo {
  public static void main(String args[]){
     for(int i=1;i<=10;i++)
     {
        System.out.print(i+" ");
     }
     }
  }
```

- **Output:-**

Practical-2
Running Java code on android studio and Demonstrating ActivityLifecycle

3.Write a java program to check whether the entered number is odd or even.

```
package com.example.lab_2;
import java.util.*;
public class OddEven {
   public static void main(String args[]){
       Scanner s=new Scanner(System.in);
       int n=s.nextInt();
       System.out.println(n%2==0?"Even":"Odd");

   }
}
```

- **Output:-**

4.Write a java program to check whether the entered number is prime or not.

```
package com.example.lab_2;

import java.util.Scanner;

public class prime {
   public static void main(String args[]) {
       Scanner s = new Scanner(System.in);
       int flag=1;
       int n = s.nextInt();
```

```
    for(int i=2;i<n/2;i++)
    {
        if(n%i==0)
        {
            flag=0;
            break;
        }
    }
    System.out.println(flag==1?"prime":"not prime");
  }
}
```

- **Output:-**

```
sampling ...
include patterns:
com\.example\.lab_2\..*
exclude patterns:
7
prime
Class transformation time: 0.0114096s for 239 classes or 4.7738912133891214E-5s per class

Process finished with exit code 0
```

5.Write a Java Program to Find Square Root of a Number Without sqrt Method.

```
package com.example.lab_2;

import java.util.Scanner;

public class sqrt {
  public static void main(String args[]) {
    Scanner s = new Scanner(System.in);
    int n = s.nextInt();
    for(int i=0;i<n;i++){
        if(i*i==n){
            System.out.println(i);
            break;
        }
```

```
    }
  }
}
```

- **Output**

```
include patterns:
com\.example\.lab_2\..*
exclude patterns:
81
9
Class transformation time: 0.0124866s for 239 classes or 5.224518828451883E-5s per class

Process finished with exit code 0
```

6.Write a Java Program to Display Even Numbers From 1 to 100

```
package com.example.lab_2;

public class OddEvenLoop {
  public static void main(String args[]){
    for(int i=2;i<=100;i+=2){
      System.out.print(i+" ");
    }
  }
}
```

- **Output:-**

```
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
com\.example\.lab_2\..*
exclude patterns:
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72

Process finished with exit code 0
```

Practical-2
Running Java code on android studio and Demonstrating ActivityLifecycle

7.Write a Java Program to Display Alternate Prime Numbers.

```
package com.example.lab_2;

public class AlternetPrime {
  public static void main(String[] args) {
     int count = 0;
     int number = 2;
     boolean flag = true;
     while (count < 10) {
        if
        (isPrime(number)) {
           if (flag) {
              System.out.println(number);
              flag = false;
           } else {
              flag = true;
           }
        }
        count++;
     }
     number++;
  }
     public static boolean isPrime ( int number){
        if (number <= 1) {
           return false;
        }
        for (int i = 2; i <= Math.sqrt(number); i++) {
           if (number % i == 0) {
              return false;
           }
        }
        return
              true;
     }
  }
```

Practical-2
Running Java code on android studio and Demonstrating ActivityLifecycle

```
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
exclude patterns:
2
5
11
17
23
Class transformation time: 0.006012093s for 172 classes or 3.495402906976744E-5s per class
```

8.Write a Java Program to Reverse a Number.

```
package com.example.lab_2;

import java.util.Scanner;

public class reverse {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter Number:"); int num = sc.nextInt(); int rev
            = 0;
      while(num!=0){
         int rem = num % 10;
         rev = rev * 10 + rem;
         num /= 10;

      }
      System.out.println("Reverse number : "+rev);
   }
}
```

```
include patterns:
com\.example\.lab_2\..*
exclude patterns:
Enter Number:
123
Reverse number : 321
Class transformation time: 0.0165417s for 235 classes or 7.039021276595745E-5s per class

Process finished with exit code 0
```

Practical-2
Running Java code on android studio and Demonstrating ActivityLifecycle

9.Write a Java Program to check whether the entered number is a Peterson Number or not.

```
package com.example.lab_2;

import java.util.Scanner;

public class peterson {
  public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     System.out.print("Enter a number: ");
     int n = sc.nextInt(); int
           temp = n, sum = 0; while
     (temp > 0) { int digit =
           temp % 10; sum +=
           factorial(digit); temp /=
           10;
     }
     if (sum == n) {
        System.out.println(n + " is a Peterson number.");
     } else {
        System.out.println(n + " is not a Peterson number."); }
     } public static int factorial(int n) {
        int fact = 1; for (int i = 2;
                     i <= n; i++) { fact *= i;
        }
        return fact;
     }
  }
```

```
sampling ...
include patterns:
com\.example\.lab_2\..*
exclude patterns:
Enter a number: 145
145 is a Peterson number.
Class transformation time: 0.0138565s for 239 classes or 5.7976987447698 75E-5s per class

Process finished with exit code 0
```

Practical-2
Running Java code on android studio and Demonstrating ActivityLifecycle

10.Write a Java Program to check whether the entered number is a Tech Number or not.

```
package com.example.lab_2;

import java.util.Scanner;

public class technum {
   public static void main(String args[])
   { int count=0;
      Scanner sc = new Scanner( System.in );
      System.out.print("Enter the number: "); int n=sc.nextInt(); while(n>0)
      {
         count++;
         n=n/10;
      }
      if(count%2==0)
         System.out.println("Number is a tech number.");
      else
         System.out.print("Number is not a tech number.");
   }
}
```

- **Output:-**

```
sampling ...
include patterns:
com\.example\.lab_2\..*
exclude patterns:
Enter the number: 2025
Number is a tech number.
Class transformation time: 0.0126262s for 235 classes or 5.372851063829787E-5s per class

Process finished with exit code 0
```

Practical-2
Running Java code on android studio and Demonstrating ActivityLifecycle

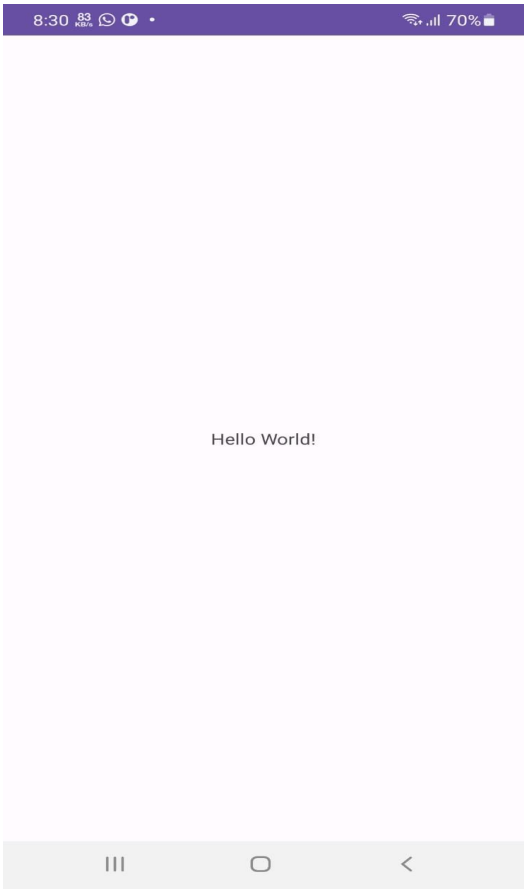
11.Demonstrating Activity Life Cycle.

```
package com.example.lab_2;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
  }
  @Override
  protected void onStart() {
     super.onStart();
     Log.d("lifecycle","onStart invoked");
  }
  @Override protected void
  onResume() {
     super.onResume();
     Log.d("lifecycle","onResume invoked");
  }
  @Override protected void
  onPause() {
     super.onPause();
     Log.d("lifecycle","onPause invoked");
  }
  @Override protected
  void onStop() {
     super.onStop();
     Log.d("lifecycle","onStop invoked");
  }
  @Override
  protected void onRestart() {
     super.onRestart();
     Log.d("lifecycle","onRestart invoked");
  }
  @Override protected void
  onDestroy() {
     super.onDestroy();
     Log.d("lifecycle","onDestroy invoked");
```

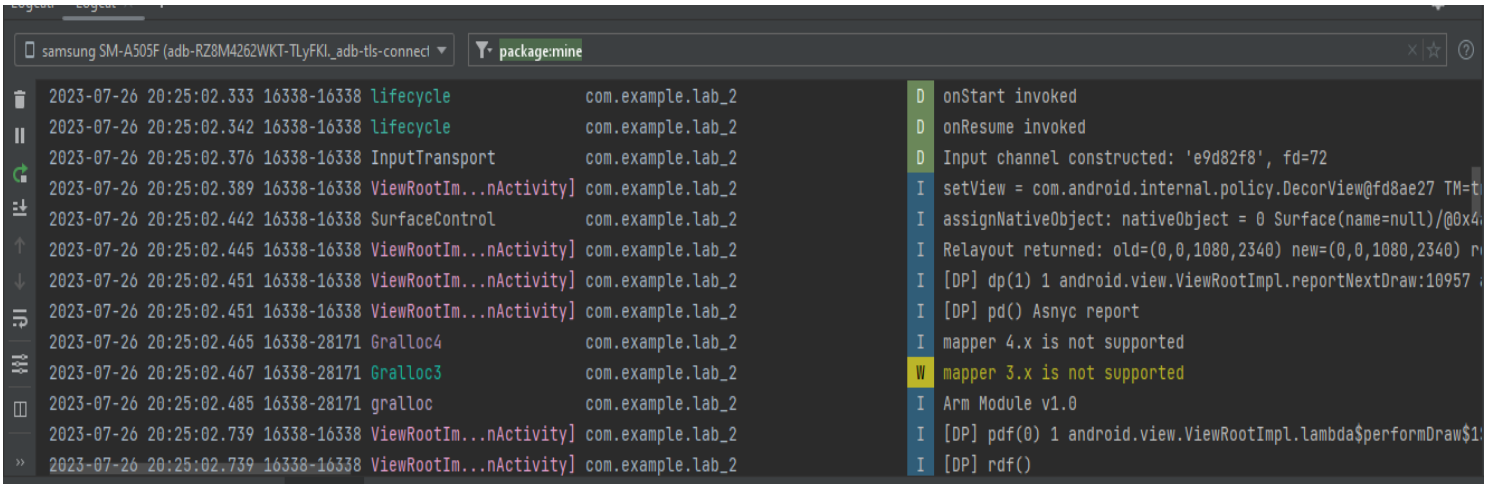
Practical-2
Running Java code on android studio and Demonstrating ActivityLifecycle

```
  }
}
```

1). Start app

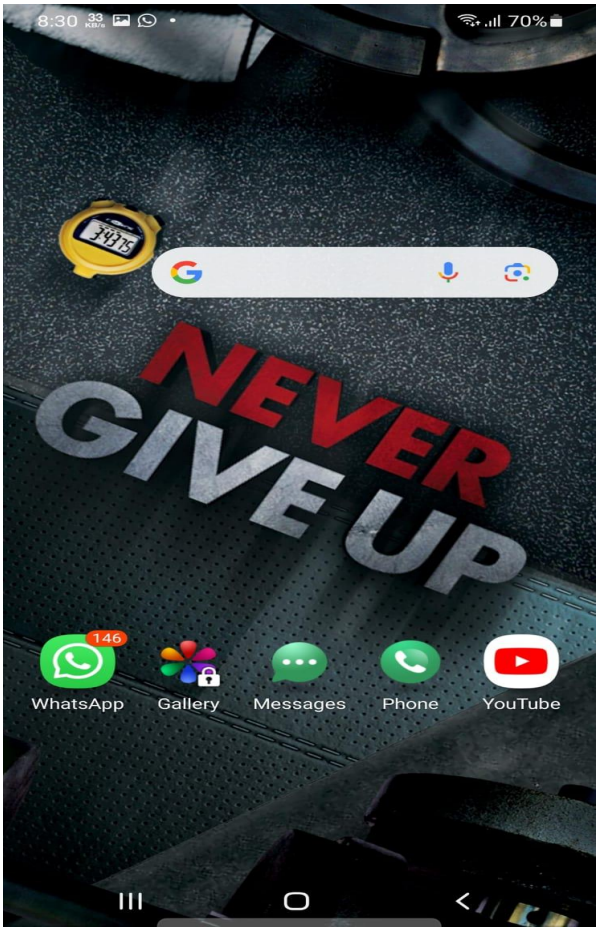


2)onStart and onResume methods are invoked.

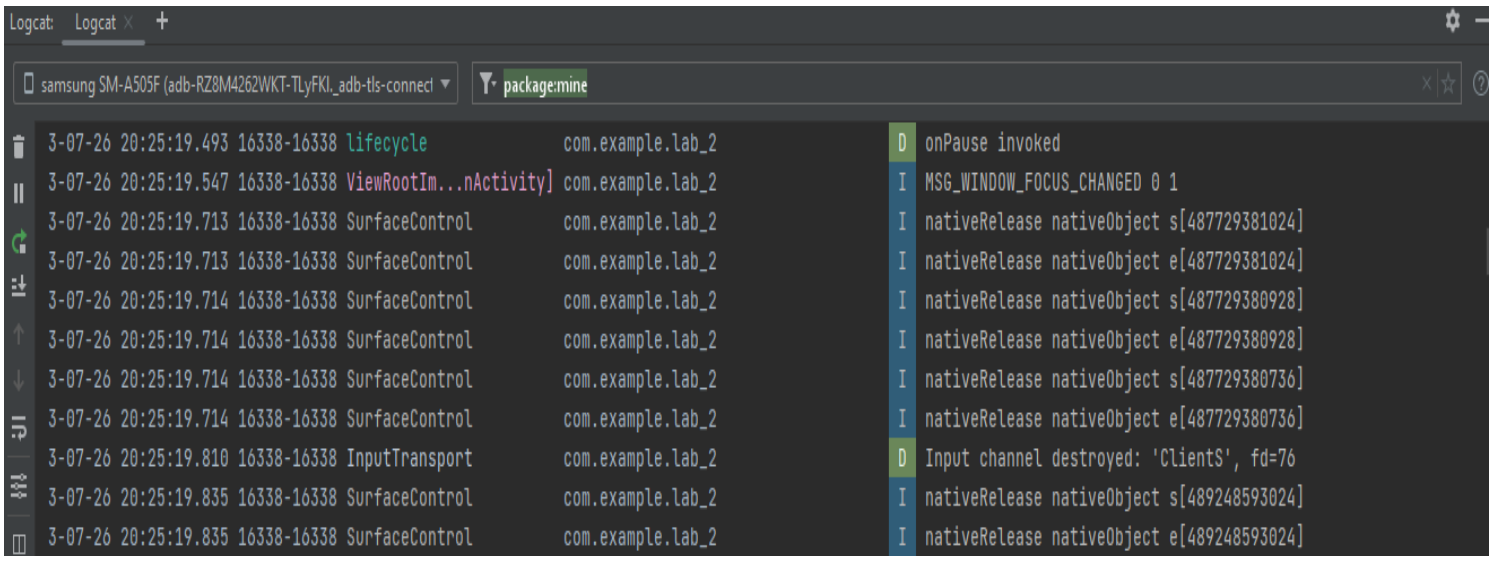


Practical-2
Running Java code on android studio and Demonstrating ActivityLifecycle

3) click HOME Button.

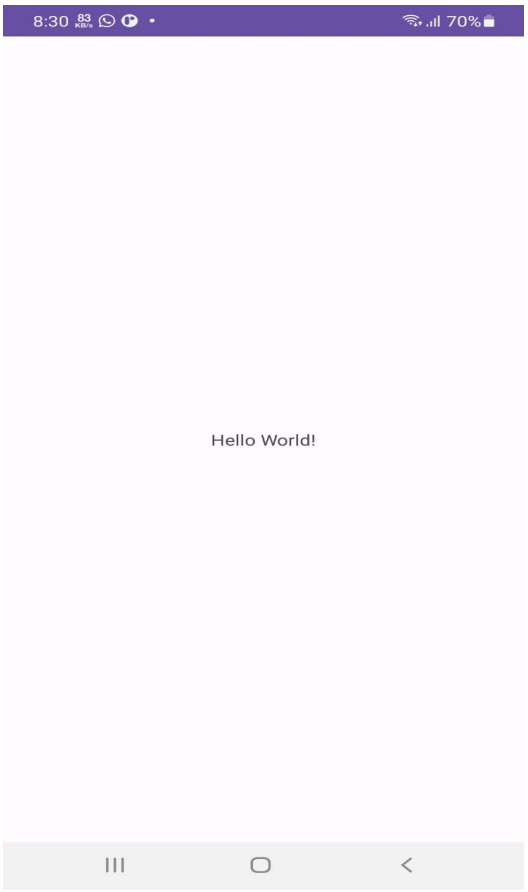


4)onPause method is invoked.

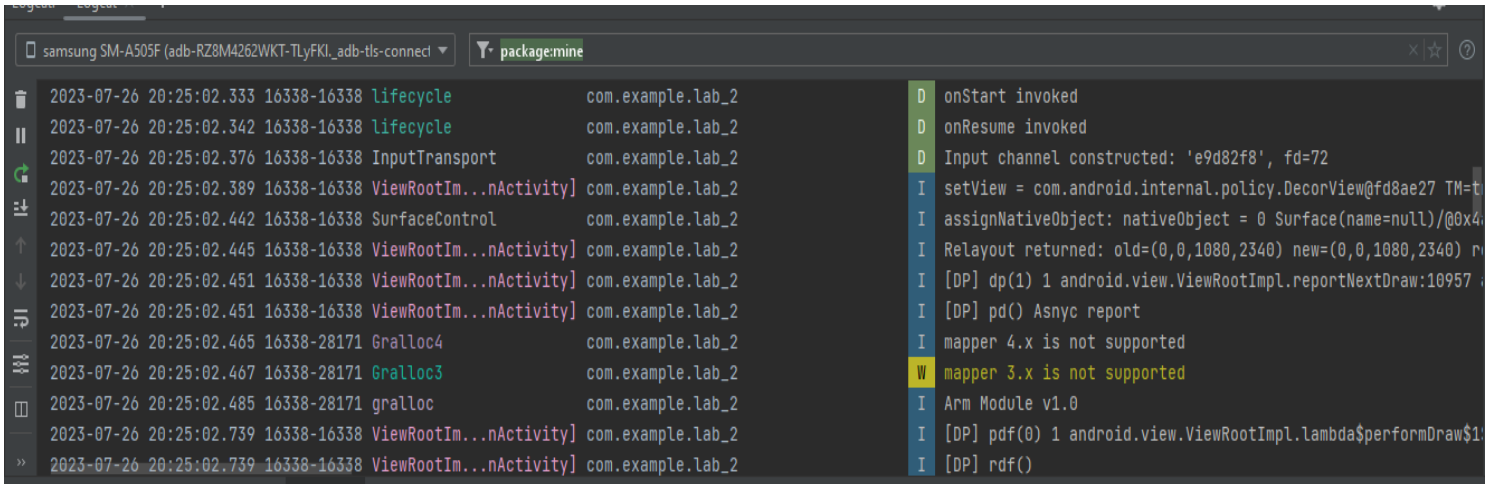


Practical-2
Running Java code on android studio and Demonstrating ActivityLifecycle

5)launch the app again



6)onStart and onResume invoked



Practical-2
Running Java code on android studio and Demonstrating ActivityLifecycle

7)Close the App

- **Onstop and onDestroy method invoked**