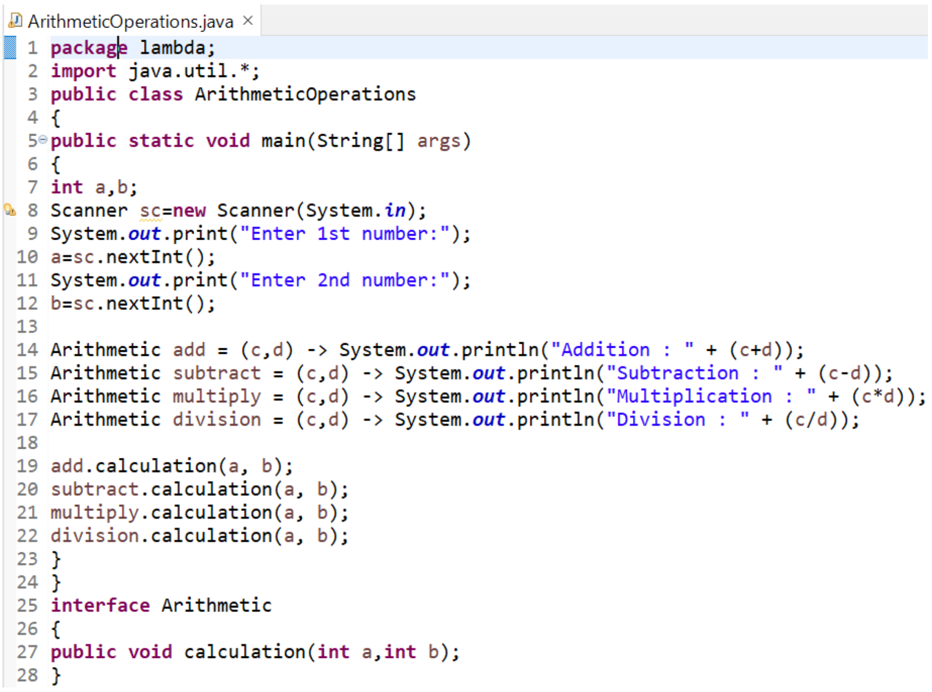
CORE JAVA

LAMBDA ASSIGNMENTS

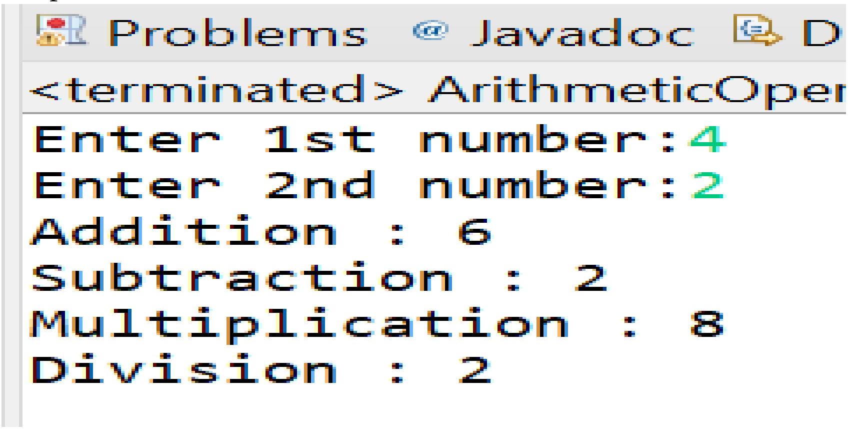
1. Write an application to perform basic arithmetic operations like add, subtract, multiply & divide. You need to define a functional interface first.
   Code: -



ArithmeticOperations.java

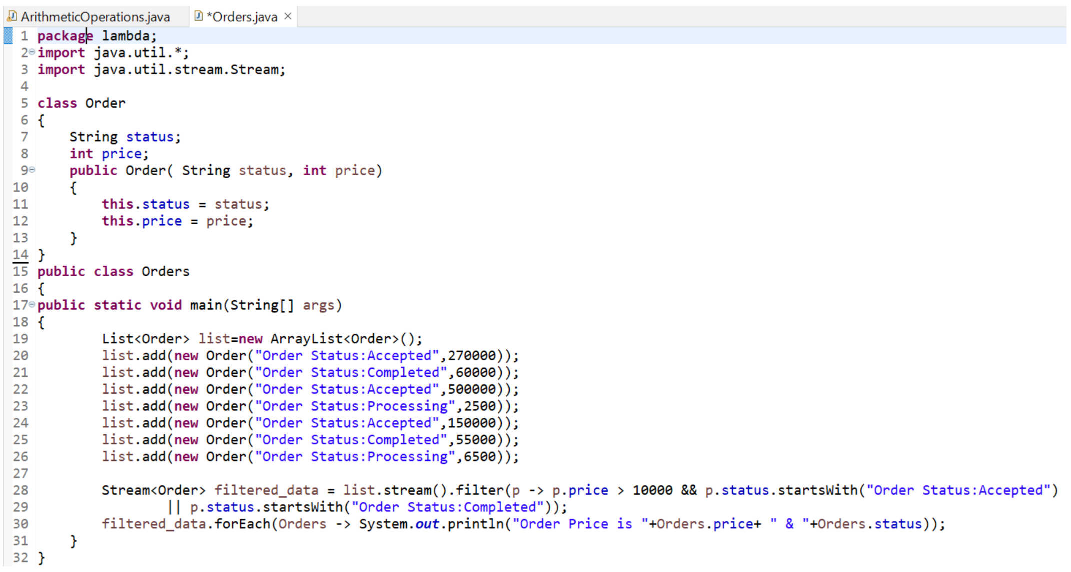
```java
package lambda;
import java.util.*;
public class ArithmeticOperations
{
public static void main(String[] args)
{
int a,b;
Scanner sc=new Scanner(System.in);
System.out.print("Enter 1st number:");
a=sc.nextInt();
System.out.print("Enter 2nd number:");
b=sc.nextInt();

Arithmetic add = (c,d) -> System.out.println("Addition : " + (c+d));
Arithmetic subtract = (c,d) -> System.out.println("Subtraction : " + (c-d));
Arithmetic multiply = (c,d) -> System.out.println("Multiplication : " + (c*d));
Arithmetic division = (c,d) -> System.out.println("Division : " + (c/d));

add.calculation(a, b);
subtract.calculation(a, b);
multiply.calculation(a, b);
division.calculation(a, b);
}
}
interface Arithmetic
{
public void calculation(int a,int b);
}
```

Output:-



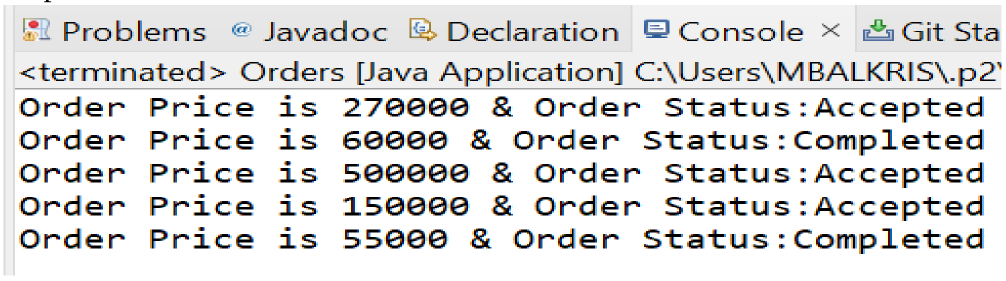
```
<terminated> ArithmeticOper
Enter 1st number:4
Enter 2nd number:2
Addition : 6
Subtraction : 2
Multiplication : 8
Division : 2
```

2. Write an application using lambda expressions to print Orders having 2 criteria implemented: 1) order price more than 10000 2) order status is ACCEPTED or COMPLETED.
   Code: -



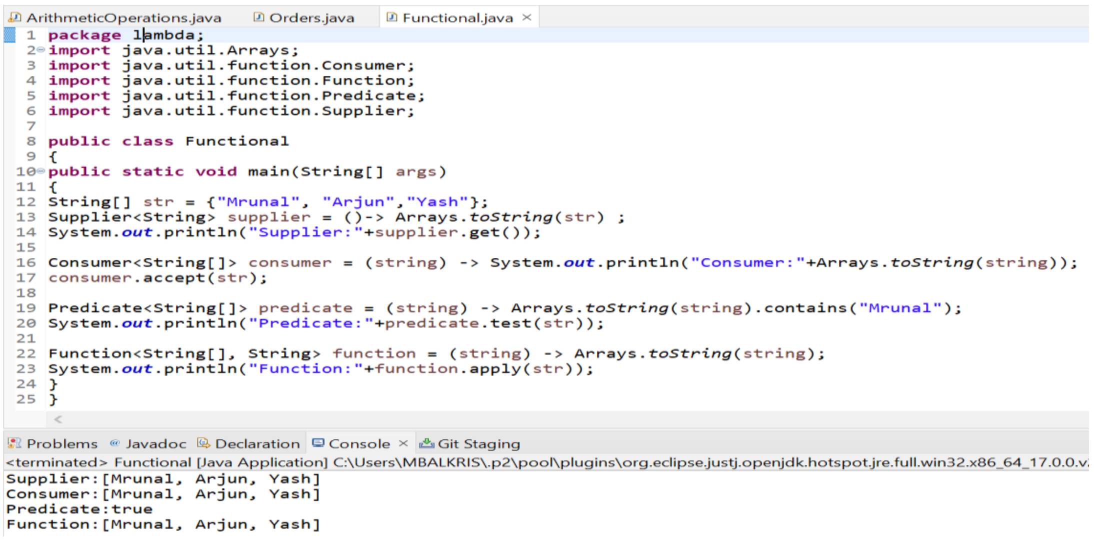
```java
package lambda;
import java.util.*;
import java.util.stream.Stream;

class Order
{
    String status;
    int price;
    public Order( String status, int price)
    {
        this.status = status;
        this.price = price;
    }
}
public class Orders
{
public static void main(String[] args)
{
        List<Order> list=new ArrayList<Order>();
        list.add(new Order("Order Status:Accepted",270000));
        list.add(new Order("Order Status:Completed",60000));
        list.add(new Order("Order Status:Accepted",500000));
        list.add(new Order("Order Status:Processing",2500));
        list.add(new Order("Order Status:Accepted",150000));
        list.add(new Order("Order Status:Completed",55000));
        list.add(new Order("Order Status:Processing",6500));

        Stream<Order> filtered_data = list.stream().filter(p -> p.price > 10000 && p.status.startsWith("Order Status:Accepted")
                || p.status.startsWith("Order Status:Completed"));
        filtered_data.forEach(Orders -> System.out.println("Order Price is "+Orders.price+ " & "+Orders.status));
    }
}
```

Output: -



```
<terminated> Orders [Java Application] C:\Users\MBALKRIS\.p2
Order Price is 270000 & Order Status:Accepted
Order Price is 60000 & Order Status:Completed
Order Price is 500000 & Order Status:Accepted
Order Price is 150000 & Order Status:Accepted
Order Price is 55000 & Order Status:Completed
```

3. Use the functional interfaces Supplier, Consumer, Predicate & Function to invoke built – in methods from Java API.
   Code: -

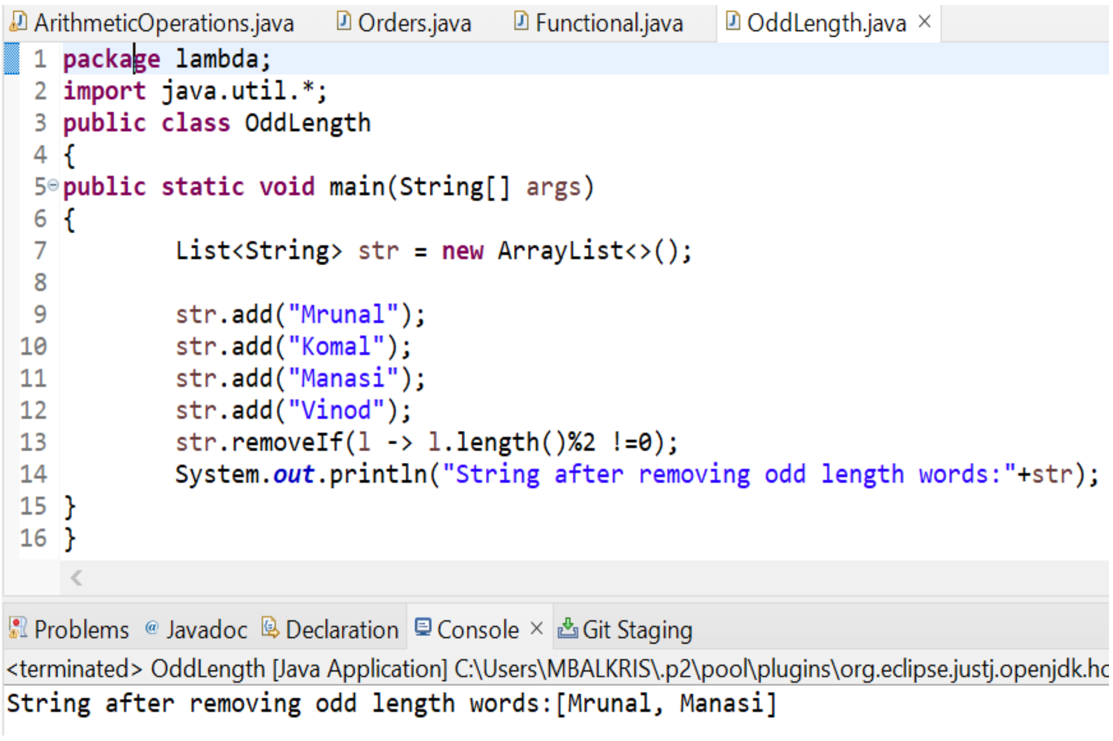


```java
package lambda;
import java.util.Arrays;
import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.function.Supplier;

public class Functional
{
public static void main(String[] args)
{
String[] str = {"Mrunal", "Arjun","Yash"};
Supplier<String> supplier = ()-> Arrays.toString(str) ;
System.out.println("Supplier:"+supplier.get());

Consumer<String[]> consumer = (string) -> System.out.println("Consumer:"+Arrays.toString(string));
consumer.accept(str);

Predicate<String[]> predicate = (string) -> Arrays.toString(string).contains("Mrunal");
System.out.println("Predicate:"+predicate.test(str));

Function<String[], String> function = (string) -> Arrays.toString(string);
System.out.println("Function:"+function.apply(str));
}
}
```

```
<terminated> Functional [Java Application] C:\Users\MBALKRIS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.0.v
Supplier:[Mrunal, Arjun, Yash]
Consumer:[Mrunal, Arjun, Yash]
Predicate:true
Function:[Mrunal, Arjun, Yash]
```

4. Remove the word that have odd lengths from the list. HINT: Use one of the new methods from JDK8. Use removeIf() method from Collection interface.

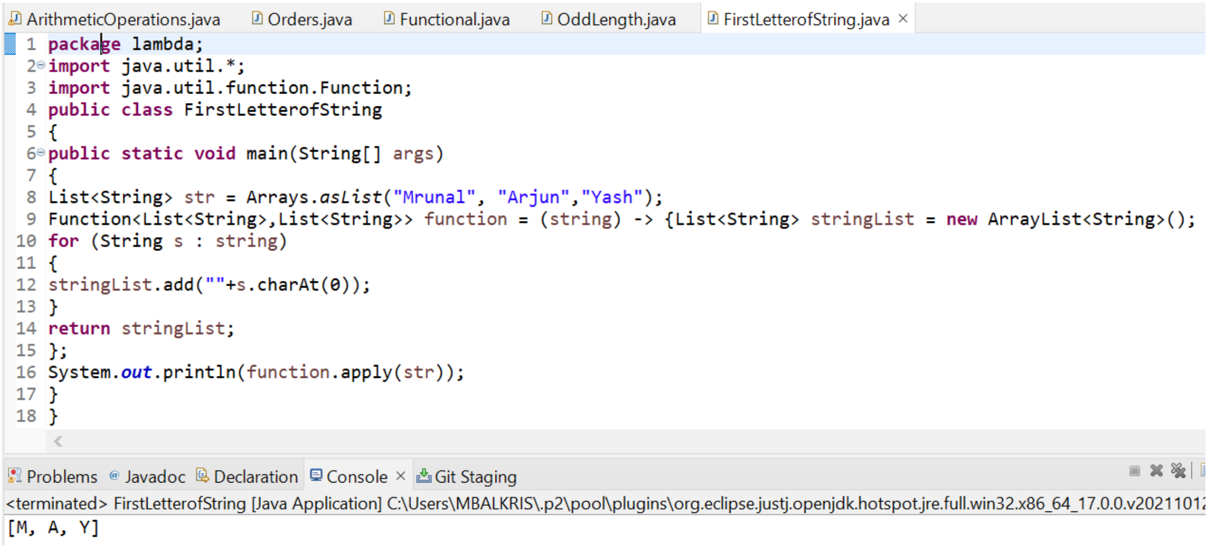
Code: -



```java
package lambda;
import java.util.*;
public class OddLength
{
public static void main(String[] args)
{
        List<String> str = new ArrayList<>();

        str.add("Mrunal");
        str.add("Komal");
        str.add("Manasi");
        str.add("Vinod");
        str.removeIf(l -> l.length()%2 !=0);
        System.out.println("String after removing odd length words:"+str);
}
}
```

```
<terminated> OddLength [Java Application] C:\Users\MBALKRIS\.p2\pool\plugins\org.eclipse.justj.openjdk.hc
String after removing odd length words:[Mrunal, Manasi]
```

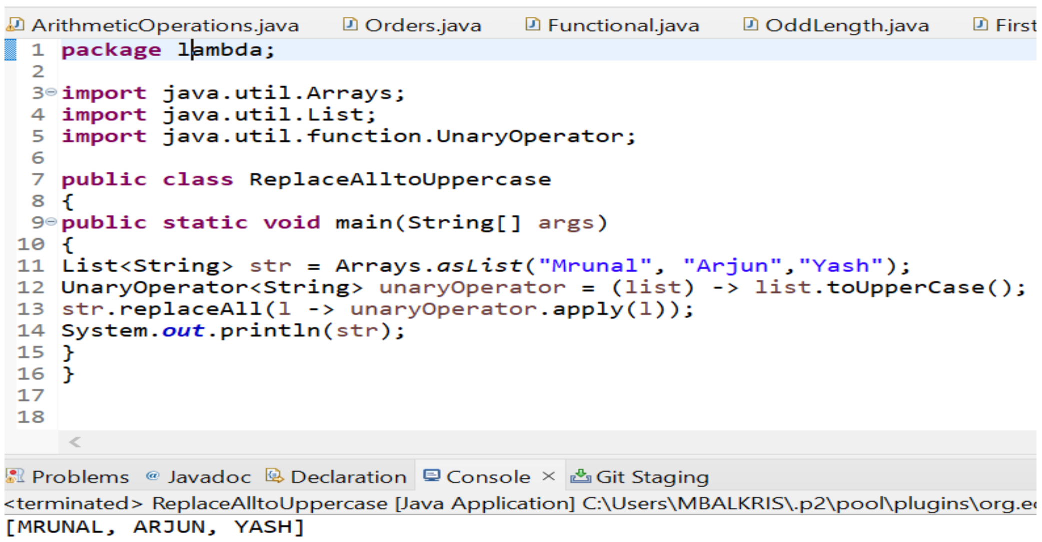
5. Create a string that consists of the first letter of each word in the list of Strings provided. HINT: Use Consumer Interface & a StringBuilder to construct the result.
   Code: -



```java
package lambda;
import java.util.*;
import java.util.function.Function;
public class FirstLetterofString
{
public static void main(String[] args)
{
List<String> str = Arrays.asList("Mrunal", "Arjun","Yash");
Function<List<String>,List<String>> function = (string) -> {List<String> stringList = new ArrayList<String>();
for (String s : string)
{
stringList.add(""+s.charAt(0));
}
return stringList;
};
System.out.println(function.apply(str));
}
}
```

```
<terminated> FirstLetterofString [Java Application] C:\Users\MBALKRIS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.0.v2021101
[M, A, Y]
```

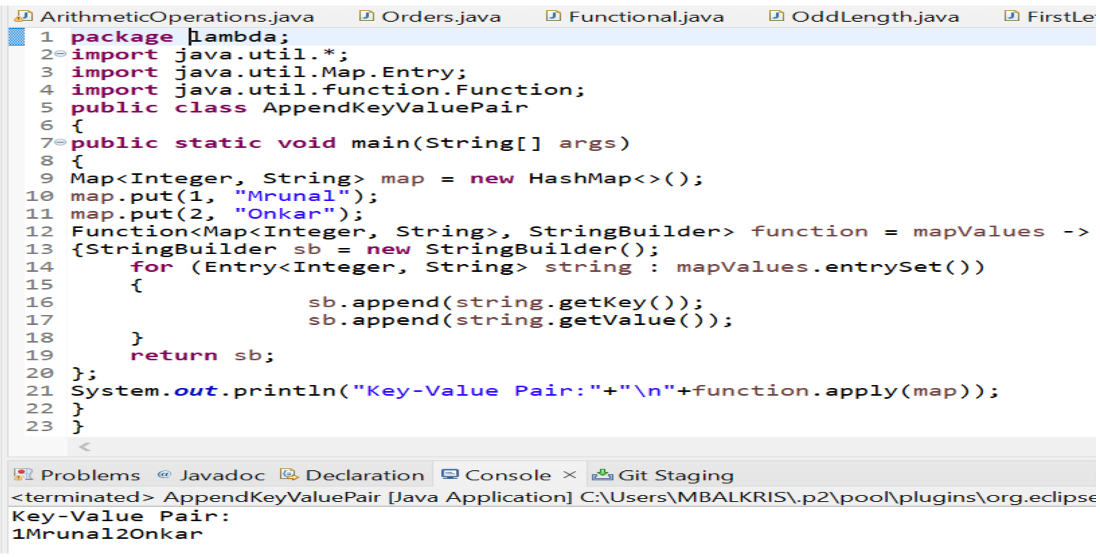
6. Replace every word in the list with its uppercase equivalent. Use replaceAll() method & UnaryOperator interface.
   Code: -



```java
package lambda;

import java.util.Arrays;
import java.util.List;
import java.util.function.UnaryOperator;

public class ReplaceAlltoUppercase
{
public static void main(String[] args)
{
List<String> str = Arrays.asList("Mrunal", "Arjun","Yash");
UnaryOperator<String> unaryOperator = (list) -> list.toUpperCase();
str.replaceAll(l -> unaryOperator.apply(l));
System.out.println(str);
}
}
```

```
<terminated> ReplaceAlltoUppercase [Java Application] C:\Users\MBALKRIS\.p2\pool\plugins\org.e
[MRUNAL, ARJUN, YASH]
```

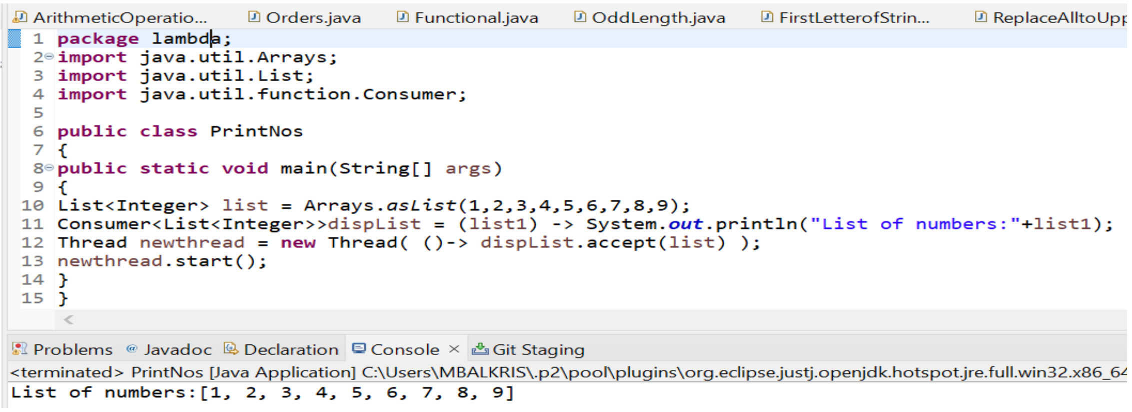
7. Convert every key- value pair of the map into a string and append them all into a single string, in iteration order. HINT: Use Map.entrySet() method & a StringBuilder to construct the result String.
   Code: -



```java
package lambda;
import java.util.*;
import java.util.Map.Entry;
import java.util.function.Function;
public class AppendKeyValuePair
{
public static void main(String[] args)
{
Map<Integer, String> map = new HashMap<>();
map.put(1, "Mrunal");
map.put(2, "Onkar");
Function<Map<Integer, String>, StringBuilder> function = mapValues ->
{StringBuilder sb = new StringBuilder();
    for (Entry<Integer, String> string : mapValues.entrySet())
    {
                sb.append(string.getKey());
                sb.append(string.getValue());
    }
    return sb;
};
System.out.println("Key-Value Pair:"+"\n"+function.apply(map));
}
}
```

```
<terminated> AppendKeyValuePair [Java Application] C:\Users\MBALKRIS\.p2\pool\plugins\org.eclipse
Key-Value Pair:
1Mrunal2Onkar
```

8. Create a new thread that prints the numbers from the list. Use class Thread & interface Consumer.
   Code: -



```java
package lambda;
import java.util.Arrays;
import java.util.List;
import java.util.function.Consumer;

public class PrintNos
{
public static void main(String[] args)
{
List<Integer> list = Arrays.asList(1,2,3,4,5,6,7,8,9);
Consumer<List<Integer>>dispList = (list1) -> System.out.println("List of numbers:"+list1);
Thread newthread = new Thread( ()-> dispList.accept(list) );
newthread.start();
}
}
```

```
<terminated> PrintNos [Java Application] C:\Users\MBALKRIS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64
List of numbers:[1, 2, 3, 4, 5, 6, 7, 8, 9]
```