

Java Generics

Problem Statement 1 : Write a Java Program to demonstrate a Generic Class.

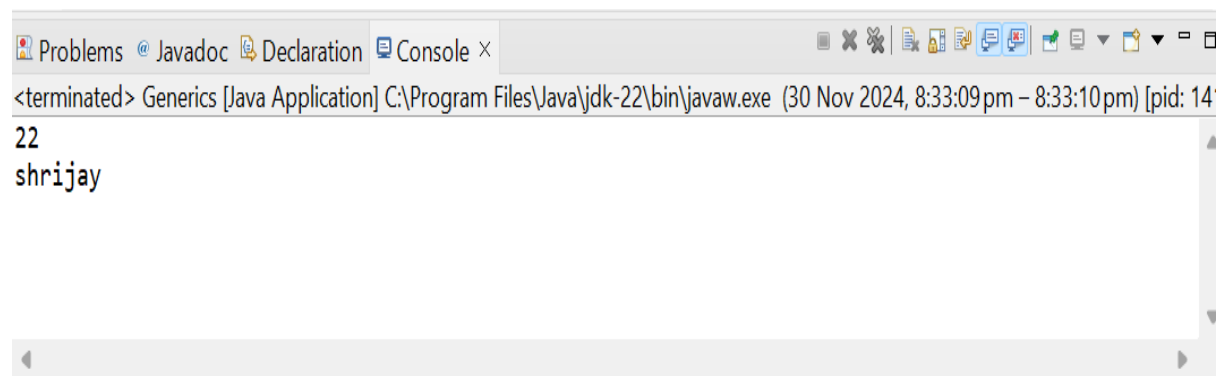
Code :

```
package com.shree;

class shree<T>{
    T obj;
    shree(T obj){
        this.obj = obj;
    }
    public T getObj() {
        return this.obj;
    }
}

public class Generics {
    public static void main(String[] args) {
        shree<Integer> obj1 = new shree<>(22);
        System.out.println(obj1.getObj());
        shree<String> obj2 = new shree<>("shrijay");
        System.out.println(obj2.getObj());
    }
}
```

Output :

A screenshot of a Java IDE's console window. The window has a title bar with tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The 'Console' tab is active, showing the output of the program. The output consists of two lines: '22' and 'shrijay'. The window title bar also includes the file path 'C:\Program Files\Java\jdk-22\bin\javaw.exe' and the timestamp '(30 Nov 2024, 8:33:09 pm - 8:33:10 pm) [pid: 14...]'.

Problem Statement 2 : Write a Java Program to demonstrate Generic Methods.

Code :

```
package com.shree;

public class GenericMethods {

    static void display()
    {
        System.out.println("generic method exmaple");
    }

    static <T> void gdisplay (T e)
    {
        System.out.println(e.getClass().getName() + " = " + e);
    }

    public static void main(String[] args) {

        display();

        gdisplay(1);

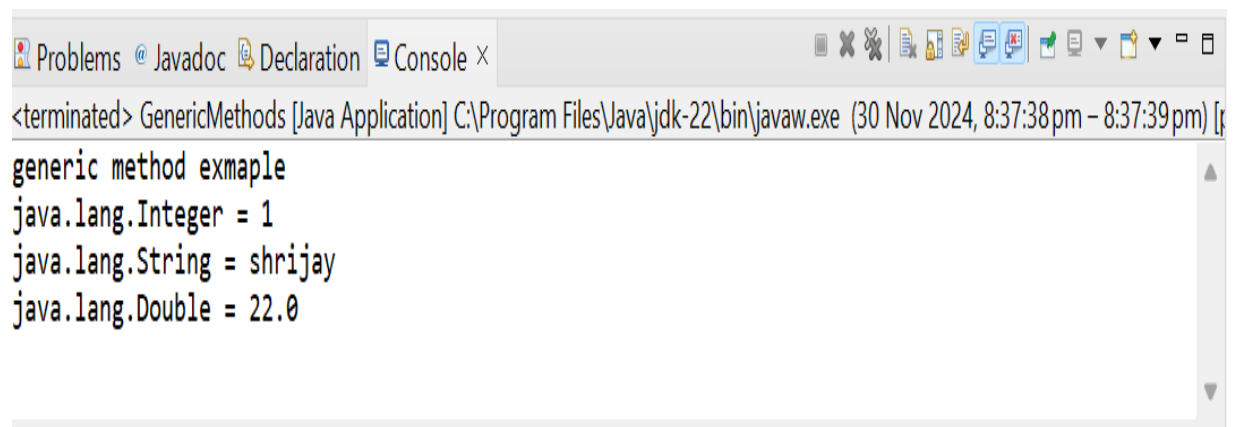
        gdisplay("shrijay");

        gdisplay(22.0);

    }

}
```

Output :

A screenshot of an IDE's console window. The title bar shows 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console text shows the program's execution: it first prints 'generic method exmaple', then 'java.lang.Integer = 1', then 'java.lang.String = shrijay', and finally 'java.lang.Double = 22.0'. The window title indicates the application is 'GenericMethods [Java Application]' running on 'C:\Program Files\Java\jdk-22\bin\javaw.exe' on '30 Nov 2024, 8:37:38 pm - 8:37:39 pm'.

Problem Statement 3 : Write a Java Program to demonstrate Wildcards in Java Generics.

Code :

```
package com.shree;

import java.util.*;

public class wildcards {

    private static double sum(List<? extends Number> list) {

        double sum = 0.0;

        for (Number i : list) {

            sum = sum + i.doubleValue();

        }

        return sum;

    }

    private static void show(List<? super Integer> list) {

        list.forEach((x) -> {

            System.out.print(x + " ");

        });

    }

    private static void print(List<?> list) {

        for(Object obj : list) {

            System.out.print(obj + " ");

        }System.out.println();

    }

    public static void main(String[] args) {

        System.out.println("Upper Bounded : ");

        List<Integer> list1 = Arrays.asList(4, 2, 7, 5, 1, 9);

        System.out.println("List 1 Sum : " + sum(list1));

        List<Double> list2 = Arrays.asList(4.7, 2.4, 7.3, 5.4, 1.5, 9.2);

        System.out.println("List 2 Sum : " + sum(list2));

        System.out.println("\nLower Bounded : ");

    }

}
```

```

        List<Integer> list3 = Arrays.asList(4, 2, 7, 5, 1, 9);

        System.out.println("Only Classes With Integer Superclass will be Accepted :");

        show(list3);

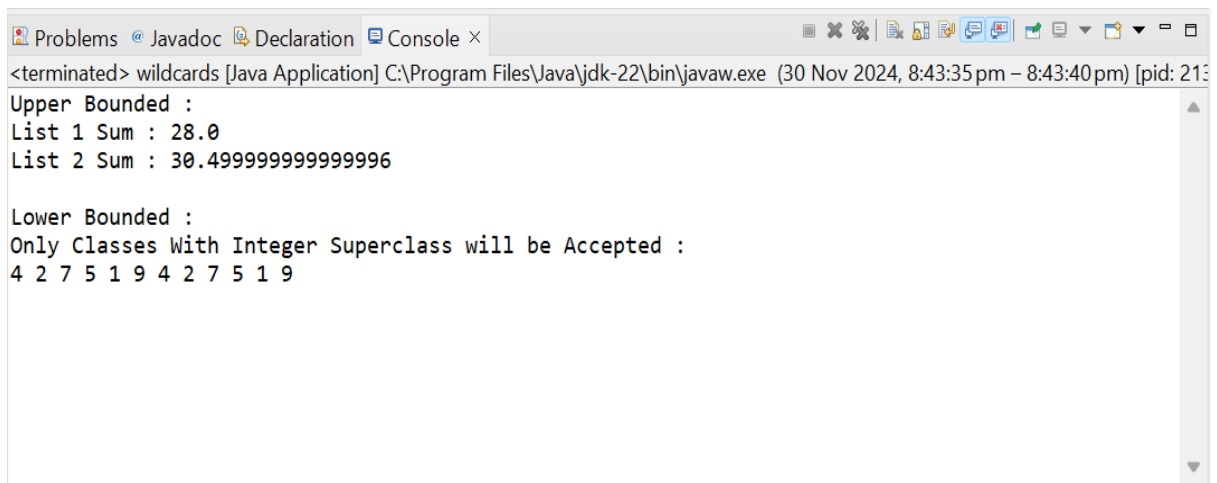
        print(list1);

    }

}

```

Output :



The screenshot shows a Java IDE console window with the following output:

```

<terminated> wildcards [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (30 Nov 2024, 8:43:35 pm – 8:43:40 pm) [pid: 213]
Upper Bounded :
List 1 Sum : 28.0
List 2 Sum : 30.499999999999996

Lower Bounded :
Only Classes With Integer Superclass will be Accepted :
4 2 7 5 1 9 4 2 7 5 1 9

```

List Interface

Problem Statement 1 : Write a Java program to create List containing list of items of type String and use for- --each loop to print the items of the list.

Code :

```
package com.shree;

import java.util.*;

public class ListEx {

    public static void main(String[] args) {

        ArrayList<String>list=new ArrayList<String>();

        list.add("Shree");

        list.add("Reyna");

        list.add("Sage");

        list.add("Sabine");

        System.out.println(list);

        System.out.println("Traversing list through for each loop");

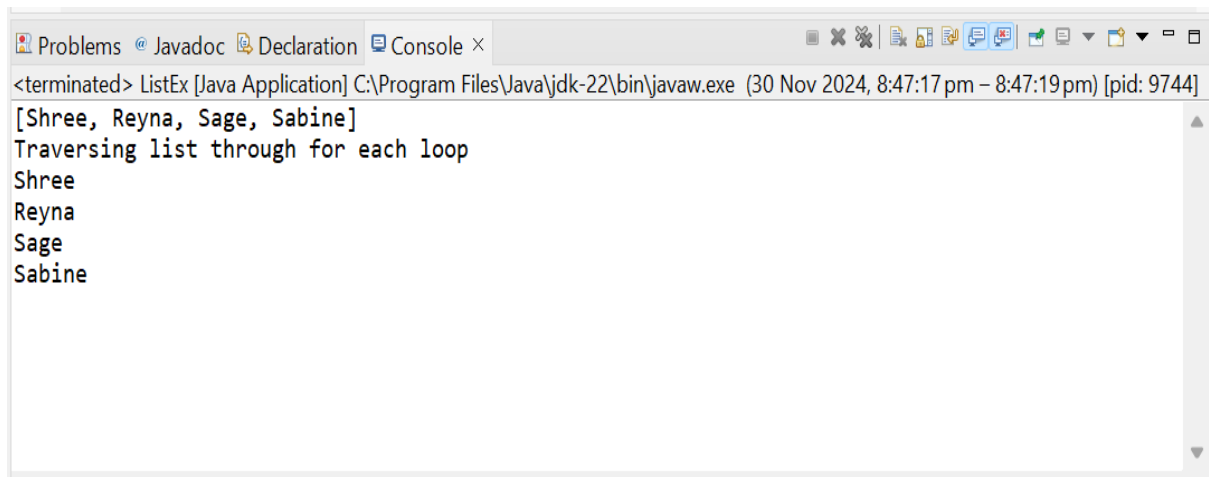
        for(String subject:list)

            System.out.println(subject);

    }

}
```

Output :

A screenshot of a Java IDE's console window. The window has a title bar with tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The 'Console' tab is active, showing the output of a Java application. The output text is: '<terminated> ListEx [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (30 Nov 2024, 8:47:17 pm – 8:47:19 pm) [pid: 9744] [Shree, Reyna, Sage, Sabine] Traversing list through for each loop Shree Reyna Sage Sabine'. The text is displayed in a monospaced font on a light background. The console window has standard OS window controls (minimize, maximize, close) and a scrollbar on the right side.

```
<terminated> ListEx [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (30 Nov 2024, 8:47:17 pm – 8:47:19 pm) [pid: 9744]
[Shree, Reyna, Sage, Sabine]
Traversing list through for each loop
Shree
Reyna
Sage
Sabine
```

Problem Statement 2 : Write a Java program to create List containing list of items and use ListIterator interface to print items present in the list. Also print the list in reverse/ backward direction.

Code :

```
package com.shree;

import java.util.*;

public class ListEx {

    public static void main(String[] args) {

        List<String> mylist = new ArrayList<String>();

        mylist.add("Shree");

        mylist.add("Reyna");

        mylist.add("Sage");

        mylist.add("Sabine");

        mylist.add("Klara");

        System.out.println("Traversing through iterator");

        System.out.println("Original List:");

        Iterator itr=mylist.iterator();

        while(itr.hasNext()) {

            System.out.println(itr.next());

        }

        Collections.reverse(mylist);

        System.out.println();

        System.out.println("Reversed List:");

        Iterator itr1=mylist.iterator();

        while(itr1.hasNext()) {

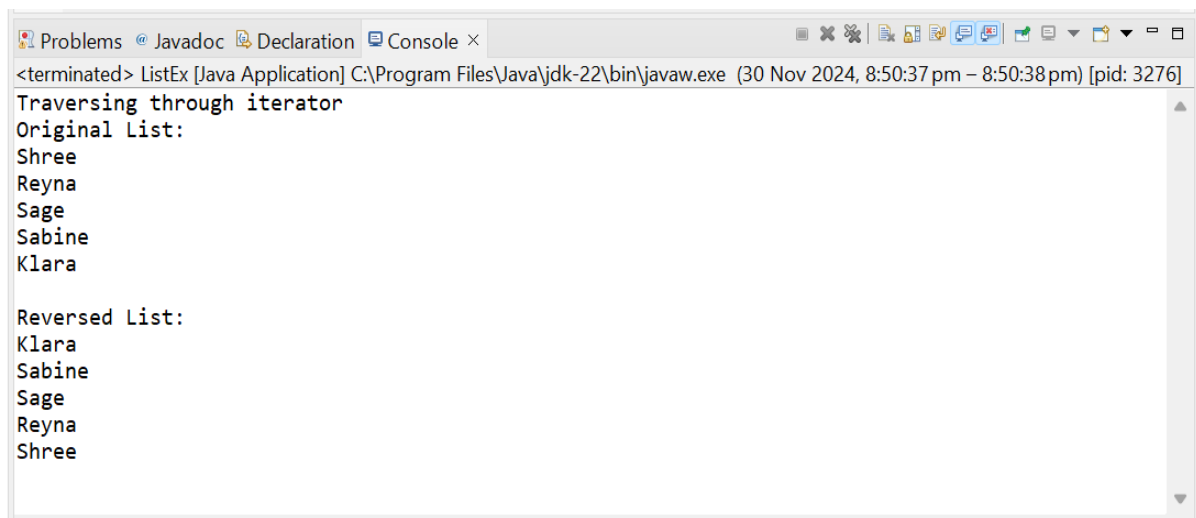
            System.out.println(itr1.next());

        }

    }

}
```

Output :



```
<terminated> ListEx [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (30 Nov 2024, 8:50:37 pm – 8:50:38 pm) [pid: 3276]
Traversing through iterator
Original List:
Shree
Reyna
Sage
Sabine
Klara

Reversed List:
Klara
Sabine
Sage
Reyna
Shree
```

Set Interface

Problem Statement 1 : Write a Java program to create a Set containing list of items of type String and print the items in the list using Iterator interface. Also print the list in reverse/backward direction.

Code :

```
package com.shree;

import java.util.*;

public class SetEx {

    public static void main(String[] args) {

        List<String> mylist = new ArrayList<String>();

        mylist.add("Shree");

        mylist.add("Reyna");

        mylist.add("Sage");

        mylist.add("Sabine");

        System.out.println("Original list ");

        Iterator<String> itr=mylist.iterator();

        while(itr.hasNext()){

            System.out.println(itr.next());

        }

        Collections.reverse(mylist);

        System.out.println(" ");

        System.out.println("reversed list ");

        Iterator<String> itr1=mylist.iterator();

        while(itr1.hasNext()){

            System.out.println(itr1.next());

        }

    }

}
```


Output :

```
<terminated> SetEx [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (30 Nov 2024, 9:25:26 pm – 9:25:27 pm) [pid: 10020]
Original list
Shree
Reyna
Sage
Sabine
|
reversed list
Sabine
Sage
Reyna
Shree
```

Problem Statement 2 : Write a Java program using Set interface containing list of items and perform the following operations :

- a. Add items in the set.
- b. Insert items of one set in to other set.
- c. Remove items from the set .
- d. Search the specified item in the set.

Code :

```
package com.shree;

import java.util.*;

public class SetEx {

    public static void main(String[] args) {

        Set<Integer> s = new LinkedHashSet<Integer>();

        s.add(69);

        s.add(57);

        s.add(10);

        s.add(18);

        s.add(90);

        s.add(151);

        Set<Integer> s1 = new LinkedHashSet<Integer>();

        s1.add(70);

        s1.add(35);

        s.addAll(s1);

        System.out.println("Set1: " + s);

        System.out.println("Set2: " + s1);

        System.out.println();

        System.out.println("After Adding set2 into set1: " + s);

        s.remove(10);

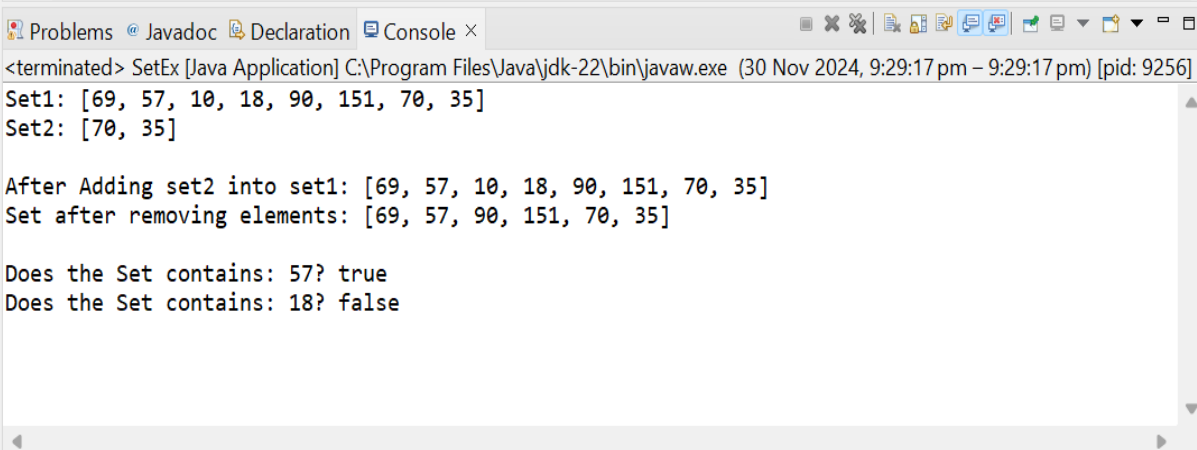
        s.remove(18);

        System.out.println("Set after removing elements: " + s);

        System.out.println();
```

```
        System.out.println("Does the Set contains: 57? "+ s.contains(57));  
        System.out.println("Does the Set contains: 18? " + s.contains(18));  
    }  
}
```

Output :



```
<terminated> SetEx [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (30 Nov 2024, 9:29:17 pm - 9:29:17 pm) [pid: 9256]  
Set1: [69, 57, 10, 18, 90, 151, 70, 35]  
Set2: [70, 35]  
  
After Adding set2 into set1: [69, 57, 10, 18, 90, 151, 70, 35]  
Set after removing elements: [69, 57, 90, 151, 70, 35]  
  
Does the Set contains: 57? true  
Does the Set contains: 18? false
```

Map Interface

Problem Statement 1 : Write a Java program using Map interface containing list of items having keys and associated values and perform the following operations :

- a. Add items in the map.
- b. Remove items from the map.
- c. Search specific key from the map.
- d. Get value of the specified key.
- e. Insert map elements of one map in to other map.
- f. Print all keys and values of the map.

Code :

```
package com.shree;

import java.util.*;

public class MapEx {

    public static void main(String[] args) {

        Map<Integer, String> map = new HashMap<>();

        map.put(1, "Shree");
        map.put(2, "Sage");
        map.put(3, "Reyna");
        map.put(4, "Sabine");
        map.put(5, "Klara");

        System.out.println();

        Map<Integer, String> map1 = new HashMap<>();

        map1.put(6, "Amir");
        map1.put(7, "Raze");
        map1.put(8, "yassine");

        System.out.println("Map 1");

        for (Map.Entry<Integer, String> e : map1.entrySet())

            System.out.println(e.getKey() + " " + e.getValue());

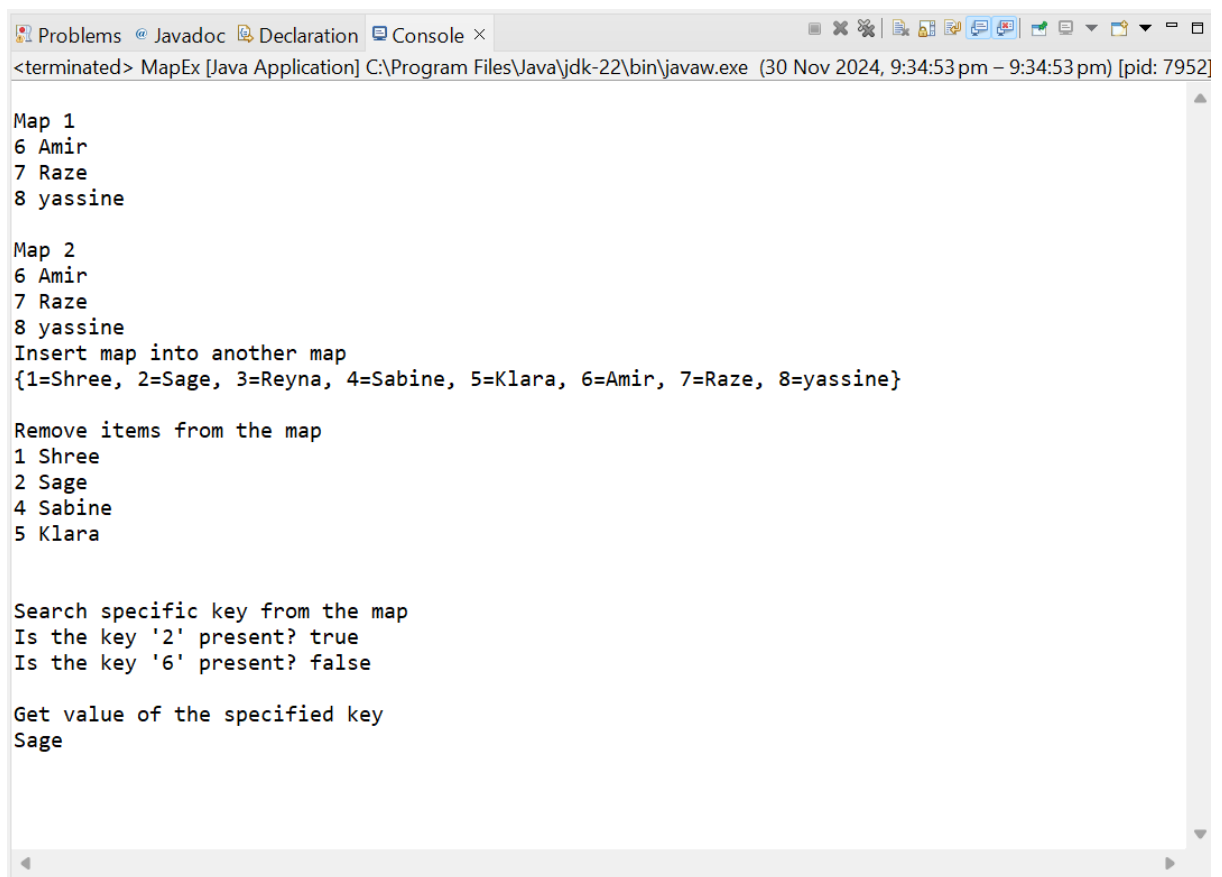
        System.out.println();
    }
}
```

```

        System.out.println("Map 2");
        for (Map.Entry<Integer, String> e : map1.entrySet())
            System.out.println(e.getKey() + " " + e.getValue());
        System.out.println("Insert map into another map");
        Map<Integer, String> map2 = new HashMap<>();
        map2.putAll(map);
        map2.putAll(map1);
        System.out.println(map2);
        System.out.println();
        System.out.println("Remove items from the map");
        map.remove((3));
        for (Map.Entry<Integer, String> e : map.entrySet())
            System.out.println(e.getKey() + " " + e.getValue());
        System.out.println();
        System.out.println();
        System.out.println("Search specific key from the map");
        System.out.println("Is the key '2' present? " +
            map.containsKey(2));
        System.out.println("Is the key '6' present? " +
            map.containsKey(6));
        System.out.println();
        System.out.println("Get value of the specified key");
        String val = (String)map.get(2);
        System.out.println(val);
        System.out.println();
    }
}

```

Output :



The screenshot shows a Java IDE window with a console tab. The console output displays the execution of a MapEx application. It starts with a terminated status, followed by the creation of Map 1 and Map 2, both containing the same three entries. Then, Map 1 is inserted into another map, resulting in a combined set of eight entries. Next, items are removed from the map, leaving five entries. Finally, a search is performed for keys '2' and '6', and the value for key '2' is retrieved.

```
<terminated> MapEx [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (30 Nov 2024, 9:34:53 pm – 9:34:53 pm) [pid: 7952]

Map 1
6 Amir
7 Raze
8 yassine

Map 2
6 Amir
7 Raze
8 yassine
Insert map into another map
{1=Shree, 2=Sage, 3=Reyna, 4=Sabine, 5=Klara, 6=Amir, 7=Raze, 8=yassine}

Remove items from the map
1 Shree
2 Sage
4 Sabine
5 Klara

Search specific key from the map
Is the key '2' present? true
Is the key '6' present? false

Get value of the specified key
Sage
```

Lambda Expressions

Problem Statement 1 : Write a Java program using Lambda Expression to print “Hello World!”.

Code :

```
package com.shree;

interface HelloWorld {

    String sayHello(String name);

}

public class LambdaExp {

    public static void main(String[] args) {

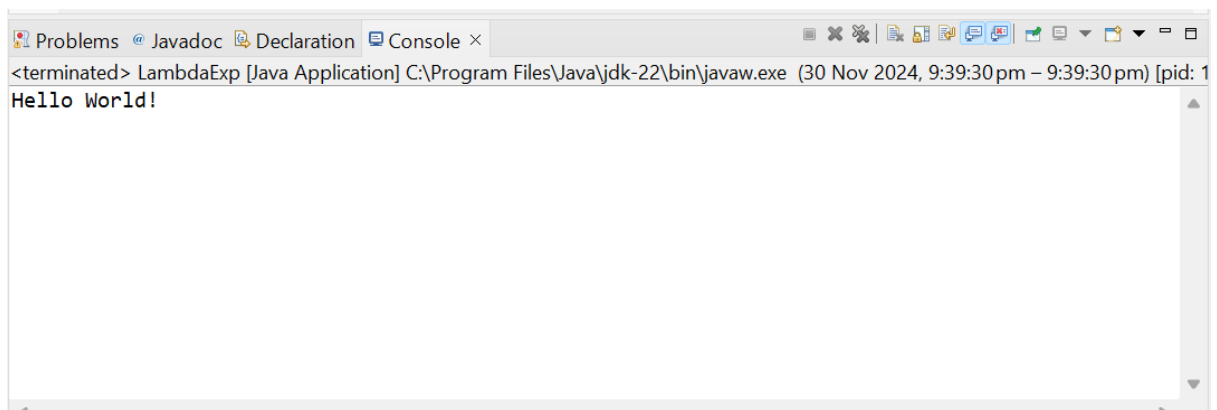
        HelloWorld helloWorld = (String name) -> { return "Hello " + name; };

        System.out.println(helloWorld.sayHello("World!"));

    }

}
```

Output :

A screenshot of an IDE's console window. The title bar shows tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console text reads: '<terminated> LambdaExp [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (30 Nov 2024, 9:39:30 pm – 9:39:30 pm) [pid: 1 Hello World!'. The output 'Hello World!' is displayed on the first line of the console area.

Problem Statement 2 : Write a Java program using Lambda Expression with single parameter.

Code :

```
package com.shree;

interface Say{

    public String say(String name);

}

public class LambdaExp {

    public static void main(String[] args) {

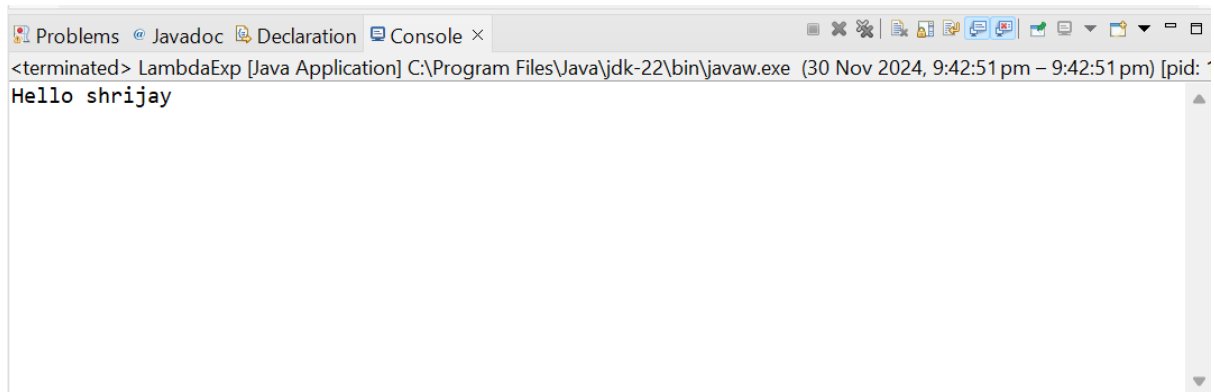
        Say s1=(name)->{return "Hello "+name;};

        System.out.println(s1.say("shrijay"));

    }

}
```

Output :



Problem Statement 3 : Write a Java program using Lambda Expression with multiple parameters to add two numbers.

Code :

```
package com.shree;

interface Add {

    int add(int a,int b);

}

public class LambdaExp {

    public static void main(String[] args) {

        Add ad1=(a,b)->(a+b);

        System.out.println("Sum: " +ad1.add(50,20));

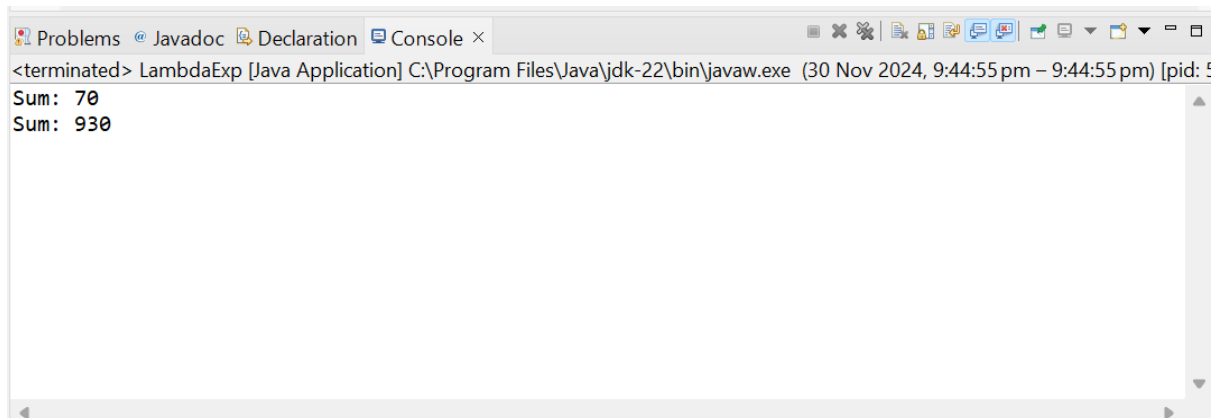
        Add ad2=(int a,int b)->(a+b);

        System.out.println("Sum: " +ad2.add(700,230));

    }

}
```

Output :

A screenshot of an IDE's console window. The window has a title bar with tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The 'Console' tab is active. The text in the console reads: '<terminated> LambdaExp [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (30 Nov 2024, 9:44:55 pm - 9:44:55 pm) [pid: 5...]' followed by two lines of output: 'Sum: 70' and 'Sum: 930'. The console window has a scrollbar on the right side.

Problem Statement 4 : Write a Java program using Lambda Expression to calculate the following:

a. Convert Fahrenheit to Celsius

Code :

```
package com.shree;

interface temp {

    public double convert(double temp);

}

public class LambdaExp {

    public static void main(String[] args) {

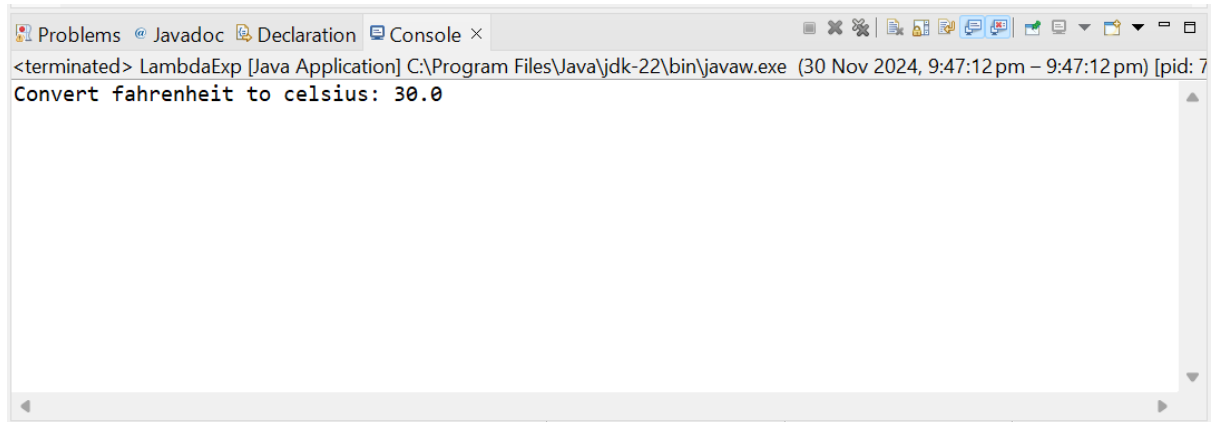
        temp t1=(double a)->{return((a-32)* 5/9);};

        System.out.print("Convert fahrenheit to celsius: "+ t1.convert(86));

    }

}
```

Output :



b. Convert Kilometers to Miles.

Code :

```
package com.shree;

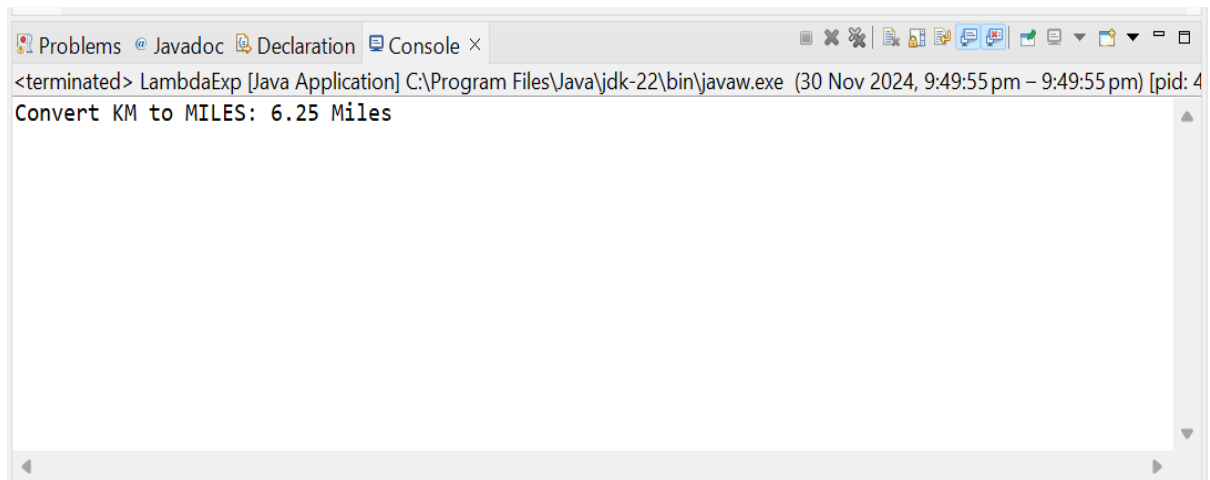
interface KmToMiles{

    public double convert(double temp);

}
```

```
public class LambdaExp {  
    public static void main(String[] args) {  
        KmToMiles kmt=(double a)->{return(a/1.6);};  
        System.out.print("Convert KM to MILES: "+ kmt.convert(10)+ " Miles");  
    }  
}
```

Output :



Problem Statement 5 : Write a Java program using Lambda Expression with or without return keyword.

Code :

```
package com.shree;

interface Add2{

    int add(int a,int b);
}

public class LambdaExp {

    public static void main(String[] args) {

        // without return keyword

        Add2 ad1=(a,b)->(a+b);

        System.out.println("Sum: " +ad1.add(43,23));


        // with return keyword

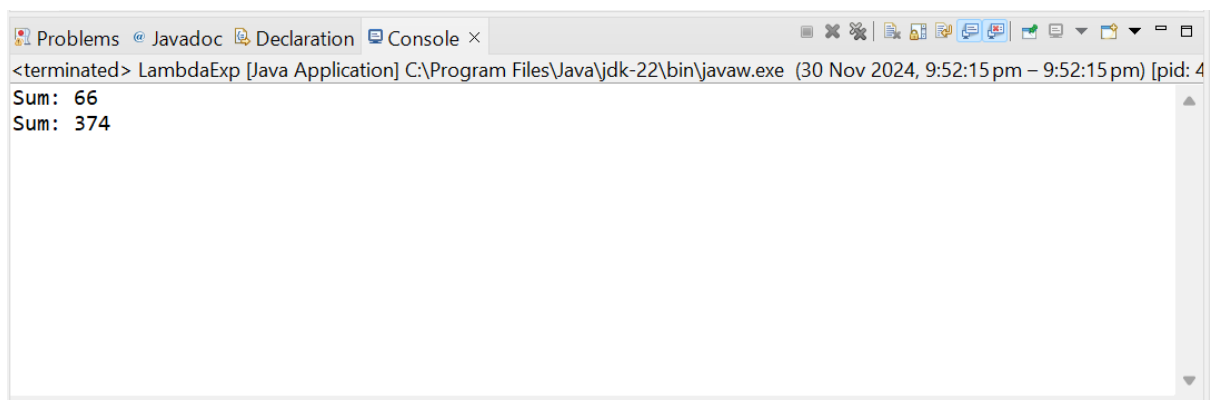
        Add2 ad2=(int a,int b)->{return (a+b);};

        System.out.println("Sum: " +ad2.add(54,320));

    }

}
```

Output :

A screenshot of an IDE's console window. The title bar shows 'Problems @ Javadoc Declaration Console x'. The console text reads: '<terminated> LambdaExp [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (30 Nov 2024, 9:52:15 pm - 9:52:15 pm) [pid: 4]'. Below this, the output is displayed: 'Sum: 66' and 'Sum: 374'. The console has a scrollbar on the right side.

Problem Statement 6 : Write a Java program using Lambda Expression to concatenate two strings.

Code :

```
package com.shree;

interface conc{

    public String concat(String a,String b);

}

public class LambdaExp {

    public static void main(String[] args) {

        conc s1 = (String a,String b)->{return (a+b)};;

        System.out.println(s1.concat("Hello"," Sage"));

    }

}
```

Output :

