

Lambda Expression

```
7.
8 class Program01 {
9
10 public static void main(String[] args
11     List<String> l1 = new ArrayList<>
12     Collections.addAll(l1, "Suresh",
13     System.out.println("Before Sort -
14
15     Collections.sort(l1); // sorting
16     System.out.println("After Sort(as
17
18     class NameComparator implements C
19         @Override
20         public int compare(String o1,
21             return o2.compareTo(o1);
22     }
23
24
25     NameComparator nameComparator = n
26
```

Name	Date modified	Type
Program01\$1NameComparator.class	08-08-2025 09:13 AM	CLASS File
Program01.class	08-08-2025 09:13 AM	CLASS File

```
19     Comparator<String> nameComparator = new Comparato
20         @Override
21         public int compare(String o1, String o2) {
22             return o2.compareTo(o1); // desc order
23         }
24     };
25
26     // Collections.sort(l1, nameComparator); // desc
27
28     // Anonymous object of Anonymous class
29     Collections.sort(l1, new Comparator<String>() {
30         @Override
31         public int compare(String o1, String o2) {
32             return o2.compareTo(o1); // desc order
33         }
34     });
35     System.out.println("After Sort(desc) -> " + l1);
36 }
37
38
```

Name	
Program01.class	0
Program01\$1.class	0

```
import java.util.ArrayList;

public class Program01 {

    public static void main(String[] args) {
        List<String> l1 = new ArrayList<>();
        Collections.addAll(l1, "Suresh", "Mukesh", "A
        System.out.println("Before Sort -> " + l1);

        Collections.sort(l1); // sorting on natural o
        System.out.println("After Sort(asc) -> " + l1

        Collections.sort(l1, (s1, s2) -> s2.compareTo
        System.out.println("After Sort(desc) -> " + l1
    }
}
```

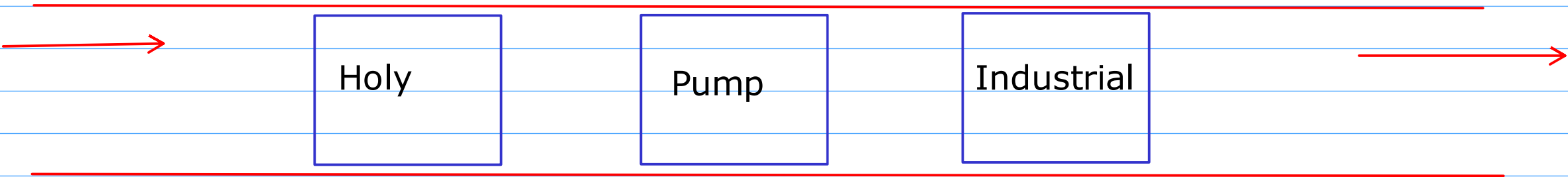
Name	
Program01.class	01

.class file is not created|

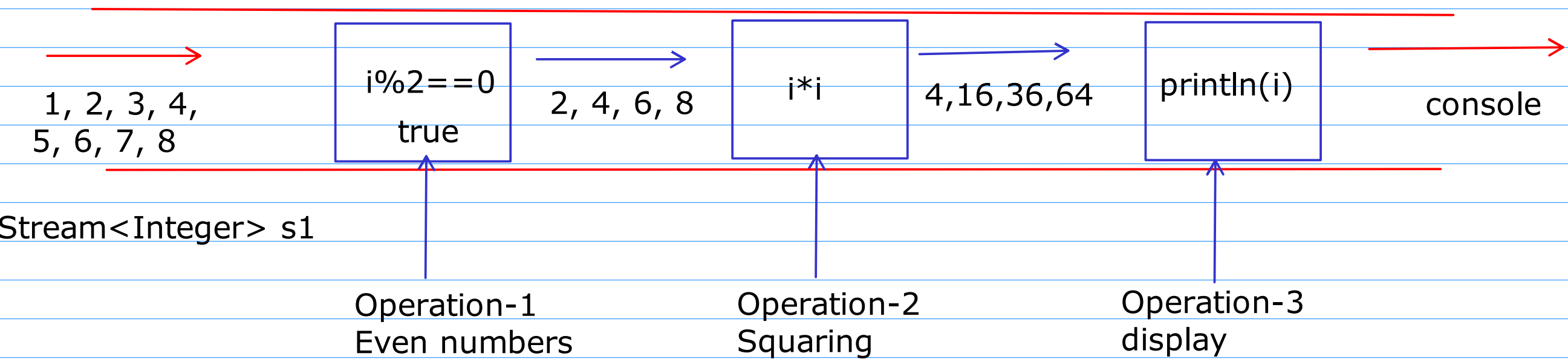
JVM instruction  
invokedynamic

- invokeSpecial -> Constructor
- invokeStatic -> Static methods
- invokeVirtual -> non static methods
- invokeInterface -> methods of interface
- invokedynamic -> lambda expression

Stream



Pipeline of operations



- Stream Object -> immutable
- Any operation on the stream will create a new Stream
- A stream once processed cannot be used again

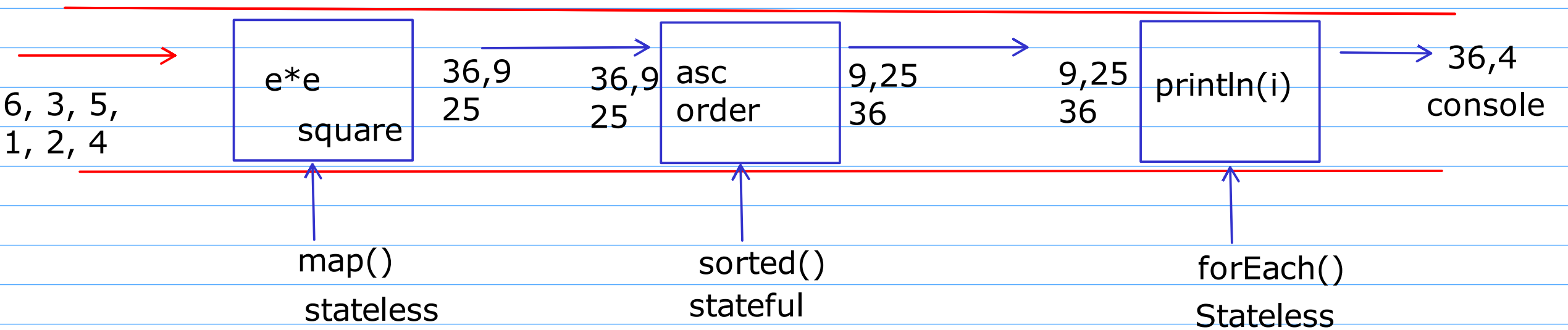
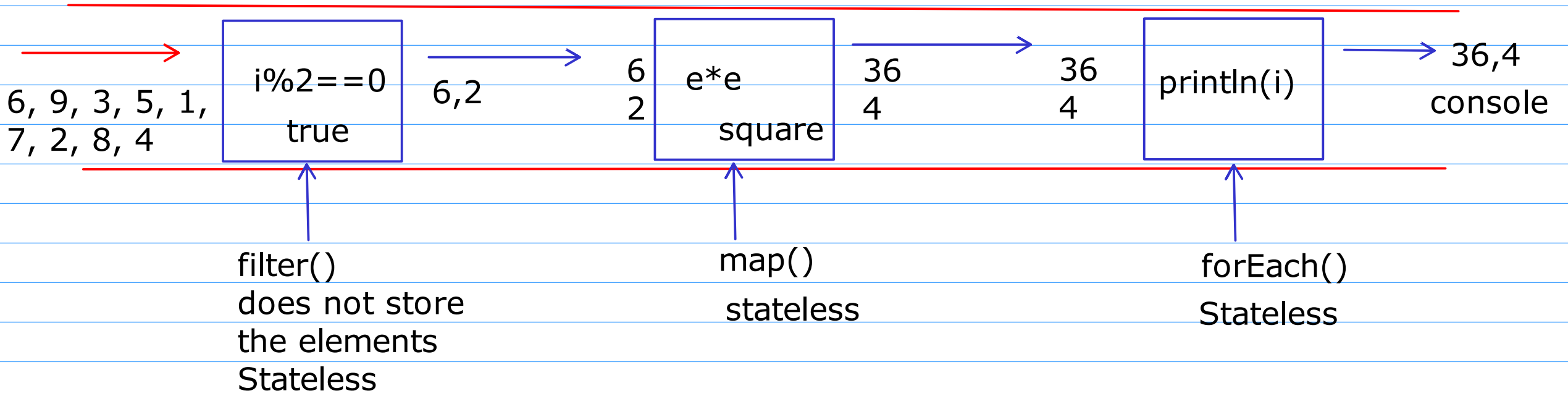
Operation on the stream are of two types

- 1. Intermediate
  - Operations that generates new Stream
- 2. Terminal Operation
  - that produces the result

Unless we provide a terminal operation the stream is not processed  
Streams are lazy evaluated

Stateless  
Stateful

Stream interface  
of()  
filter()  
count()  
limit()  
skip()  
map()  
sorted()  
sorted(Cmparator)  
distinct()



File IO

MultiThreading

Annotation

Reflection

Nested class

Enum

Optional

recursion

80 hrs -> 20 days

80hrs > 21 days

LinkedList