Week 1:

Exercise 1:

1. I4 defined twice
2. x: 50
   B class m1
   Y: 60
   B class m2
   A class m3
3. 10
   10
   20
4. 30 40
   10 20
   5 6
   10 20
5. 30 40
   10 20
   30 10
   5 40
   6 20
   5 6
6. x: 30
   x: 10
   Y: 40
   Y: 20
7. 10
   20
   30
   40

   10
   20
   50
   60

Exercise 2:

a.

0 null 0.0
0 null 0.0

b.

111 ankit 5000.0
112 sumit 6000.0

c.
111 ankit 5000.0
112 sumit 6000.0

d.
hello n
hello m

e.
hello a
10

f.
5
hello a

g.
111 ankit java 0.0
112 sumit java 6000.0

h.
11 ankit java 0.0
112 sumit java 6000.0

i. method is invoked

j. 10

h. hello java

super

a.
black
white

b.
eating...
barking...

c.
animal is created
dog is created

d.
animal is created
dog is created

e.
1 ankit 45000.0

Exercise 3 Polymorphism :
1.
B class m1
B class m2
A class m3 string-param
B m3 float, int param
B m3 float, int param
B class m1
B class m2
A class m3 no-param
A class m3 string-param
B m3 float, int param

2.
B class m1
B class m2
A class m3
A m4 no param
B m4 String param
B class m1
B class m2
A class m3
A m4 no param
B m4 String param

3.
B float arg
B float arg
B float arg

B float arg
B float arg
B float arg

4.
B float arg
B float arg

B float arg

B float arg
B float arg
B float arg

5.
A int arg
B char arg
A int arg

A int arg
B char arg
A int arg

6. A int arg
B char arg
A int arg

A int arg
B char arg
A int arg

7. B float arg
B long arg
B float arg

B float arg
B long arg
B float arg

8.
B String arg
A object arg

B String arg
A object arg

9.
B object arg
B object arg

B object arg
B object arg

Overriding:
a.

```
class Example {
   void m1() {}
}

class Sample extends Example {
   void m1() {}
}
```

b.

```
class Example {
   static void m1() {}
}

class Sample extends Example {
   static void m1() {}
}
```

c.

```
class Example {
   void m1() {}
}

class Sample {
   private void m1() {}
}
```

Accessibility Modifiers:
1.
a: 10
b: 20
c: 30
d: 40

2. error: a has private access in Example

3. error : a has private access in class A

Exercise 4:
a.

```
abstract class Example {
   abstract void m1();
   abstract int m2();
   abstract void m3(String str);
   abstract int m4();
```

b.
error: missing method body, or declare abstract

c.
Example is not abstract and does not override abstract method m1() in Example

d.
Cannot instantiate abstract class

e.
error: missing method body, or declare abstract

1.
Reb bus engine capacity is 40 KMPH
Breaks applied bus stopped
Bus will run on 6 wheels
volvo bus engine capacity is 110 KMPH
Breaks applied bus stopped
Bus will run on 6 wheels

2.
a. need to use default

```
interface Shape {
   default void findArea() {
      System.out.println("Shape area");
   }
}

public class Myjava implements Shape{
   public static void main(String args[]){
      Shape s = new Myjava();
      s.findArea();
   }
}
```

b. all methods should be public

c. all methods should be public

program output
1.
Reb bus engine capacity is 40 KMPH
Breaks applied bus stopped
volvo bus engine capacity is 110 KMPH
Breaks applied bus stopped

2.
I, Lion eats non-veg
I, Lion sleeps in caves
I, Rabbit eats veg
I, Rabbit sleeps in bushes

String Handling :
1.
S2: abc
S3: abc
S4: 'bbc
false
true
S6: abady
S7: b.
S8: abc
S9: b

2. StringIndexOutOfBoundsException: offset -2, count 4, length 6

s12 chars: cdeE
s12 length: 4

s15 chars:
s15 length: 0

s16 chars:
s16 length: 0

3.
s1: null
s1 length: null

S4: null54
S4 length: 6

5.
S1 length: 0
S2 length: 2
S3 length: 4
S4 length: 0
S5 length: 0
S6 length: 1
S7 length: 2
Is S1 empty: true
5
3
3
2

6.
S1: Hi
S1: Hi
S2: Hi
S2: Hi
S3: null
a: 1234

Programs:

```
import java.io.*;

class PalindromeCheck {
   public static boolean checkPalindrome(String inputStr) {
      String reversedStr = "";
      boolean result = false;

      for (int i = inputStr.length() - 1; i >= 0; i--) {
         reversedStr = reversedStr + inputStr.charAt(i);
      }

      if (inputStr.equals(reversedStr)) {
         result = true;
      }
      return result;
   }

   public static void main(String[] args) {
```

```
        String inputString = "lol";
        inputString = inputString.toLowerCase();
        boolean isPalindrome = checkPalindrome(inputString);
        System.out.println(isPalindrome);
    }
}
```

2.

```
public class MyClass {

    public String revstr(String st) {
        char[] s = st.toCharArray();
        for (int i = 0, j = s.length - 1; i < s.length / 2; i++, j--) {
            char temp = s[i];
            s[i] = s[j];
            s[j] = temp;
        }
        return new String(s);
    }

    public static void main(String[] args) {
        MyClass myClass = new MyClass();
        String reversed = myClass.revstr("abcd");
        System.out.println(reversed);
    }
}
```

3.

```
String name= "Aman";
char letter = 'A';
int count = 0;

for (int i = 0; i <name.length(); i++) {
    if (name.charAt(i) == letter) {
        count++;
    }
}
System.out.println(count);
```

4.

```
import java.util.HashMap;
import java.util.Map;

public class CharacterCount {
    public static void main(String[] args) {
```

```
        String input = "abcbcbcadabcc";
        Map<Character, Integer> charCountMap = new HashMap<>();

        for (char ch : input.toCharArray()) {
            charCountMap.put(ch, charCountMap.getOrDefault(ch, 0) + 1);
        }

        for (Map.Entry<Character, Integer> entry : charCountMap.entrySet()) {
            System.out.println(entry.getKey() + " " + entry.getValue());
        }
    }
}
```

5.

```
public class StringSearch {
    public static void main(String[] args) {
        String str1 = "Aman";
        String str2 = "My name is Aman";

        if (str2.contains(str1)) {
            System.out.println(str1 + " is present in " + str2);
        } else {
            System.out.println(str1 + " is not present in " + str2);
        }
    }
}
```

6. Java checks the pool to see if an identical string already exists

7.An immutable class is a class whose objects cannot be modified after they are created, we use final key word

8. String is immutable, StringBuffer is mutable and thread-safe, StringBuilder is mutable but not thread-safe.

JAVA 8 Features:
3.Output:
a.Hello, this is default method
Work is worship

b.Hello, this is default method from SayableExtra
Work is worship

Lambda Expressions
4.Output:
a.Drawing 10
b. Drawing 10
c. 30
300

5.

```
import java.util.function.Predicate;

public class PrimeNumberChecker {
   public static void main(String[] args) {
      int numToCheck = 17;

      Predicate<Integer> isPrime = num -> {
         if (num <= 1) {
            return false;
         }
         for (int i = 2; i <= Math.sqrt(num); i++) {
            if (num % i == 0) {
               return false;
            }
         }
         return true;
      };

      if (isPrime.test(numToCheck)) {
         System.out.println(numToCheck + " is a prime number.");
      } else {
         System.out.println(numToCheck + " is not a prime number.");
      }
   }
}
```

6.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

class Worker {
   private String workerName;

   public Worker(String workerName) {
```

```
        this.workerName = workerName;
    }

    public String getWorkerName() {
        return workerName;
    }
}

public class WorkerSorter {
    public static void main(String[] args) {
        List<Worker> workers = new ArrayList<>();
        workers.add(new Worker("Aman"));
        workers.add(new Worker("Rahul"));


        Comparator<Worker> nameComparator = (w1, w2) ->
w2.getWorkerName().compareTo(w1.getWorkerName());

        Collections.sort(workers, nameComparator);

        for (Worker worker : workers) {
            System.out.println(worker.getWorkerName());
        }
    }
}

7.
public class FibonacciCalculator {
    public static void main(String[] args) {
        int targetValue = 10;

        Thread fibonacciThread = new Thread(() -> {
            long result = calculateFibonacci(targetValue);
            System.out.println("Fibonacci value for " + targetValue + ": " + result);
        });

        fibonacciThread.start();
    }

    private static long calculateFibonacci(int n) {
        if (n <= 0) {
            return 0;
        } else if (n == 1) {
            return 1;
```

```
    } else {
      long fib = 0, fibPrev = 1, fibTemp;
      for (int i = 2; i <= n; i++) {
        fibTemp = fib;
        fib = fib + fibPrev;
        fibPrev = fibTemp;
      }
      return fib;
    }
  }
}
```

Optional Class:

```
import java.util.HashMap;
import java.util.Optional;

public class PhoneBook {

  private static final HashMap<String, String> PHONE_NUMBERS = new HashMap<String,
String>() {
    {
      put("Jos de Vos", "016/161616");
      put("An de Toekan", "016/161617");
      put("Kris de Vis", "016/161618");
    }
  };

  private HashMap<String, String> phoneBookEntries = PHONE_NUMBERS;

  PhoneBook() { }

  public HashMap<String, String> getPhoneBookEntries() {
    return phoneBookEntries;
  }

  public Optional<String> findPhoneNumberByName(String name) {
    String phoneNumber = phoneBookEntries.get(name);
    return Optional.ofNullable(phoneNumber);
  }

  public Optional<String> findNameByPhoneNumber(String phoneNumber) {
    for (HashMap.Entry<String, String> entry : phoneBookEntries.entrySet()) {
      if (entry.getValue().equals(phoneNumber)) {
        return Optional.of(entry.getKey());
```

```
        }
      }
      return Optional.empty();
   }

   @Override
   public String toString() {
      System.out.println("Hello from PhoneBook's toString method");
      return "PhoneBook{" +
            "phoneBookEntries=" + phoneBookEntries +
            '}';
   }
}

##
import java.util.Optional;

public class PhoneBookCrawler {

   private PhoneBook2 phoneBook;

   public PhoneBookCrawler(PhoneBook2 phoneBook) {
      this.phoneBook = phoneBook;
   }

   public String findPhoneNumberByNameAndPunishIfNothingFound(String name) {
      Optional<String> phoneNumber = phoneBook.findPhoneNumberByName(name);
      return phoneNumber.orElseThrow(() -> new IllegalArgumentException("No phone number
found"));
   }

   public String findPhoneNumberByNameAndPrintPhoneBookIfNothingFound(String name) {
      return null;
   }

   public String findPhoneNumberByNameOrNameByPhoneNumber(String name, String
phoneNumber) {
      return null;
   }

   public PhoneBook2 getPhoneBook() {
      return phoneBook;
   }
}
```

Streams:

```
import java.util.Arrays;
import java.util.List;

class Worker {
   int id;
   String name;
   double monthlyIncome;
   boolean markedForLayoff;

   public Worker(int id, String name, double monthlyIncome) {
      this.id = id;
      this.name = name;
      this.monthlyIncome = monthlyIncome;
   }

   public void markForLayoff() {
      markedForLayoff = true;
   }


   public String toString() {
      return "Worker{id=" + id + ", name='" + name + "', monthlyIncome=" + monthlyIncome + ",
markedForLayoff=" + markedForLayoff + '}';
   }
}

public class WorkerSalaryIncrement {
   public static void main(String[] args) {
      List<Worker> workers = Arrays.asList(
            new Worker(1, "Aman", 2000),
            new Worker(2, "Aryan", 55000),
            new Worker(3, "abhi", 50001)
      );

      workers.forEach(worker -> {
         double updatedIncome = worker.monthlyIncome * 1.1;
         if (updatedIncome > 50000) worker.markForLayoff();
         worker.monthlyIncome = updatedIncome;
      });

      workers.forEach(System.out::println);
   }
```

}