ASSIGNMENT NO.2

Note: Consider the following before starting the assignment:

- A **static field** declared inside a class is called a **class-level variable**. To access this variable, use the class name and the dot operator (e.g., `Integer.MAX_VALUE`).
- A **static method** defined inside a class is called a **class-level method**. To access this method, use the class name and the dot operator (e.g., `Integer.parseInt()`).
- When accessing static members within the same class, you do not need to use the class name.

1. Working with `java.lang.Boolean`

a. Explore the Java API documentation for `java.lang.Boolean` and observe its modifiers and super types.

b. Declare a method-local variable `status` of type `boolean` with the value `true` and convert it to a `String` using the `toString` method. (Hint: Use `Boolean.toString(Boolean)`).

Sol:-
```
class HelloWorld {

    public static void main(String[] args) {

        boolean status = true;

        String s1 = Boolean.toString(status);

        System.out.println(s1);

    }

}
```

c. Declare a method-local variable `strStatus` of type `String` with the value `"true"` and convert it to a `boolean` using the `parseBoolean` method. (Hint: Use `Boolean.parseBoolean(String)`).

Sol:-
```
class HelloWorld {

    public static void main(String[] args) {

        String s1 = "true";

        boolean status = Boolean.parseBoolean(s1);
        System.out.println(status);

    }

}
```

d. Declare a method-local variable `strStatus` of type `String` with the value `"1"` or `"0"` and attempt to convert it to a `boolean`. (Hint: `parseBoolean` method will not work as expected with `"1"` or `"0"`).

Sol:- parseBoolean does not recognise '1' or '0' and is giving false for both

to convert proper we need a logic to convert 1 to true and 0 to false

```
class HelloWorld {

    public static void main(String[] args) {

        String s1 = "1";

        if(s1.equals("1"))

        s1="true";

        else if(s1.equals("0"))

        s1="false";

        boolean status = Boolean.valueOf(s1);

        System.out.println(status);

    }

}
```

e. Declare a method-local variable `status` of type `boolean` with the value `true` and convert it to the corresponding wrapper class using `Boolean.valueOf()`. (Hint: Use `Boolean.valueOf(boolean)`).

Sol:-

```
class HelloWorld {
   public static void main(String[] args) {
      boolean b =true;
       Boolean a = Boolean.valueOf(b);
      System.out.println(a);
   }
}
```

f. Declare a method-local variable `strStatus` of type `String` with the value `"true"` and convert it to the corresponding wrapper class using `Boolean.valueOf()`. (Hint: Use `Boolean.valueOf(String)`).

Sol:-

```
class HelloWorld {
   public static void main(String[] args) {
      String b ="true";
      Boolean a = Boolean.valueOf(b);
      System.out.println(a);
   }
}
```

g. Experiment with converting a `boolean` value into other primitive types or vice versa and observe the results.

Sol:- for all it gives incompatible type conversion error

ASSIGNMENT NO.2

2. Working with `java.lang.Byte`

a. Explore the Java API documentation for `java.lang.Byte` and observe its modifiers and super types.

b. Write a program to test how many bytes are used to represent a `byte` value using the `BYTES` field. (Hint: Use `Byte.BYTES`).

Sol:-

```
class HelloWorld {

    public static void main(String[] args) {

        System.out.println(Byte.BYTES);

    }

}
```

it uses 1 byte to store a byte

c. Write a program to find the minimum and maximum values of `byte` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Byte.MIN_VALUE` and `Byte.MAX_VALUE`).

```
class HelloWorld {

    public static void main(String[] args) {

      byte b=2;

        System.out.println(Byte.MIN_VALUE);

    }

}
```

min value is -128.

d. Declare a method-local variable `number` of type `byte` with some value and convert it to a `String` using the `toString` method. (Hint: Use `Byte.toString(byte)`).

```
class HelloWorld {

    public static void main(String[] args) {

      byte number= 123;

      String s1=Byte.toString(number);

        System.out.println(s1);

    }

}
```

e. Declare a method-local variable `strNumber` of type `String` with some value and convert it to a `byte` value using the `parseByte` method. (Hint: Use `Byte.parseByte(String)`).

```
class HelloWorld {

    public static void main(String[] args) {

      String number= "123";

      byte s1=Byte.parseByte(number);

        System.out.println(s1);

    }

}
```

f. Declare a method-local variable `strNumber` of type `String` with the value `"Ab12Cd3"` and attempt to convert it to a `byte` value. (Hint: `parseByte` method will throw a `NumberFormatException`).

```
class HelloWorld {

    public static void main(String[] args) {

      String number= "Ab12Cd3";
      byte s1=Byte.parseByte(number);

       System.out.println(s1);

    }

}
```

g. Declare a method-local variable `number` of type `byte` with some value and convert it to the corresponding wrapper class using `Byte.valueOf()`. (Hint: Use `Byte.valueOf(byte)`)

```
class HelloWorld {

    public static void main(String[] args) {

      byte number= 12;

      Byte s1=Byte.valueOf(number);

       System.out.println(s1);

    }

}

.
```

h. Declare a method-local variable `strNumber` of type `String` with some `byte` value and convert it to the corresponding wrapper class using `Byte.valueOf()`. (Hint: Use `Byte.valueOf(String)`).

```
class HelloWorld {

    public static void main(String[] args) {

      String number= "12";
      Byte s1=Byte.valueOf(number);

      System.out.println(s1);

    }

}
```

i. Experiment with converting a `byte` value into other primitive types or vice versa and observe the results

byte can be converted to other primitive data types but other data types cannot be converted without explicit type casting (narrowing)

3. Working with `java.lang.Short`

a. Explore the Java API documentation for `java.lang.Short` and observe its modifiers and super types.

b. Write a program to test how many bytes are used to represent a `short` value using the `BYTES` field. (Hint: Use `Short.BYTES`).

```
class HelloWorld {

    public static void main(String[] args) {

        System.out.println(Short.BYTES);

    }

}
```

c. Write a program to find the minimum and maximum values of `short` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Short.MIN_VALUE` and `Short.MAX_VALUE`).

```
class HelloWorld {

    public static void main(String[] args) {

        System.out.println(Short.MIN_VALUE+" "+Short.MAX_VALUE);

    }
}
```

min=-32768 max=32767

d. Declare a method-local variable `number` of type `short` with some value and convert it to a `String` using the `toString` method. (Hint: Use `Short.toString(short)`).

```
class HelloWorld {

    public static void main(String[] args) {

        short s=123;

        String a=Short.toString(s);

        System.out.println(a);

    }

}
```

e. Declare a method-local variable `strNumber` of type `String` with some value and convert it to a `short` value using the `parseShort` method. (Hint: Use `Short.parseShort(String)`).

```
class HelloWorld {

    public static void main(String[] args) {

        String s="123";

        short a=Short.parseShort(s);

        System.out.println(a);

    }

}
```

f. Declare a method-local variable `strNumber` of type `String` with the value `"Ab12Cd3"` and attempt to convert it to a `short` value. (Hint: `parseShort` method will throw a `NumberFormatException`).

it throws NUmberformatexception error

g. Declare a method-local variable `number` of type `short` with some value and convert it to the corresponding wrapper class using `Short.valueOf()`. (Hint: Use `Short.valueOf(short)`).

```
class HelloWorld {

    public static void main(String[] args) {

        short number=123;

        short a=Short.valueOf(number);

        System.out.println(a);

    }

}
```

h. Declare a method-local variable `strNumber` of type `String` with some `short` value and convert it to the corresponding wrapper class using `Short.valueOf()`. (Hint: Use `Short.valueOf(String)`).

```
class HelloWorld {

    public static void main(String[] args) {

        String number="123";

        short a=Short.valueOf(number);

        System.out.println(a);

    }

}
```

i. Experiment with converting a `short` value into other primitive types or vice versa and observe the results.

short can be converted to int,float,double but to convert from int , double ,float to short we need (narrowing)

4. Working with `java.lang.Integer`

a. Explore the Java API documentation for `java.lang.Integer` and observe its modifiers and super types.

b. Write a program to test how many bytes are used to represent an `int` value using the `BYTES` field. (Hint: Use `Integer.BYTES`).

```
class HelloWorld {

    public static void main(String[] args) {

        System.out.println(Integer.BYTES);

    }

}
```

4 bytes

c. Write a program to find the minimum and maximum values of `int` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Integer.MIN_VALUE` and `Integer.MAX_VALUE`).

```
class HelloWorld {

    public static void main(String[] args) {

        System.out.println(Integer.MIN_VALUE+" "+Integer.MAX_VALUE);

    }

}
```

d. Declare a method-local variable `number` of type `int` with some value and convert it to a `String` using the `toString` method. (Hint: Use `Integer.toString(int)`).

```
class HelloWorld {

    public static void main(String[] args) {

        int n=1234;

        String s1=Integer.toString(n);

        System.out.println(s1);

    }

}
```

e. Declare a method-local variable `strNumber` of type `String` with some value and convert it to an `int` value using the `parseInt` method. (Hint: Use `Integer.parseInt(String)`).

```
public class Day3 {



             public static void main(String[] args) {

//

         String s1 = "123";

         int i = Integer.parseInt(s1);

         System.out.println(i);

             }



}
```

f. Declare a method-local variable `strNumber` of type `String` with the value `"Ab12Cd3"` and attempt to convert it to an `int` value. (Hint: `parseInt` method will throw a `NumberFormatException`).

it throws java.lang.NumberFormatException

g. Declare a method-local variable `number` of type `int` with some value and convert it to the corresponding wrapper class using `Integer.valueOf()`. (Hint: Use `Integer.valueOf(int)`).

```
public class Day3 {



        public static void main(String[] args) {

  int number =123;

   Integer i = Integer.valueOf(number);

      System.out.println(i);

        }



}
```

h. Declare a method-local variable `strNumber` of type `String` with some integer value and convert it to the corresponding wrapper class using `Integer.valueOf()`. (Hint: Use `Integer.valueOf(String)`).

```
public class Day3 {

        public static void main(String[] args) {

                String s1 = "123";

                int i = Integer.valueOf(s1);

      System.out.println(i);

        }



}
```

ASSIGNMENT NO.2

i. Declare two integer variables with values `10` and `20`, and add them using a method from the `Integer` class. (Hint: Use `Integer.sum(int, int)`).

```
public class Day3 {

            public static void main(String[] args) {

             int i=20;int j=30;

             int result = Integer.sum(i, j);



        System.out.println(result);

            }



}
```

j. Declare two integer variables with values `10` and `20`, and find the minimum and maximum values using the `Integer` class. (Hint: Use `Integer.min(int, int)` and `Integer.max(int, int)`).

```
public class Day3 {

            public static void main(String[] args) {

             int i=20;int j=30;

             int result = Integer.max(i, j);



        System.out.println(result);

            }



}
```

k. Declare an integer variable with the value `7`. Convert it to binary, octal, and hexadecimal strings using methods from the `Integer` class. (Hint: Use `Integer.toBinaryString(int)`, `Integer.toOctalString(int)`, and `Integer.toHexString(int)`).

```
public class Day3 {

        public static void main(String[] args) {

         int i=7;

         String result = Integer.toBinaryString(i);

         String r1 = Integer.toOctalString(i);

         String r2 = Integer.toHexString(i);

     System.out.println(result+" "+r1+" "+r2);

        }


}
```

l. Experiment with converting an `int` value into other primitive types or vice versa and observe the results.

int can be converted into double float long

5. Working with `java.lang.Long`

a. Explore the [Java API documentation for `java.lang.Long`]() and observe its modifiers and super types.

b. Write a program to test how many bytes are used to represent a `long` value using the `BYTES` field. (Hint: Use `Long.BYTES`).

```
public class Day3 {

        public static void main(String[] args) {

    System.out.println(Long.BYTES);

        }
}
```

it gives output 8 bytes

c. Write a program to find the minimum and maximum values of `long` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Long.MIN_VALUE` and `Long.MAX_VALUE`).

```
public class Day3 {

        public static void main(String[] args) {

    System.out.println(Long.MAX_VALUE+" "+Long.MIN_VALUE);

        }


}
```

```
9223372036854775807 -9223372036854775808
```

d. Declare a method-local variable `number` of type `long` with some value and convert it to a `String` using the `toString` method. (Hint: Use `Long.toString(long)`).

```
public class Day3 {

        public static void main(String[] args) {

                long l=123l;

                String s1=Long.toString(l);

    System.out.println(s1);

        }


}
```

e. Declare a method-local variable `strNumber` of type `String` with some value and convert it to a `long` value using the `parseLong` method. (Hint: Use `Long.parseLong(String)`).

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

//

        String s1 = "123";

        long i = Long.parseLong(s1);

        System.out.println(i);

            }



}
```

f. Declare a method-local variable `strNumber` of type `String` with the value `"Ab12Cd3"` and attempt to convert it to a `long` value. (Hint: `parseLong` method will throw a `NumberFormatException`).

it throws java.lang.NumberFormatException

g. Declare a method-local variable `number` of type `long` with some value and convert it to the corresponding wrapper class using `Long.valueOf()`. (Hint: Use `Long.valueOf(long)`).

```
public class Day3 {



            public static void main(String[] args) {

//

  long number =123l;

    Long i = Long.valueOf(number);

        System.out.println(i);

            }
```

}

h. Declare a method-local variable `strNumber` of type `String` with some `long` value and convert it to the corresponding wrapper class using `Long.valueOf()`. (Hint: Use `Long.valueOf(String)`).

```
public class Day3 {

        public static void main(String[] args) {

                String s1="123";

                long l=Long.valueOf(s1);

    System.out.println(l);

        }


}
```

i. Declare two long variables with values `1123` and `9845`, and add them using a method from the `Long` class. (Hint: Use `Long.sum(long, long)`).

```
public class Day3 {

        public static void main(String[] args) {

                long l1=112l;long l2=123l;

                long l=Long.sum(l1,l2);

    System.out.println(l);

        }


}
```

j. Declare two long variables with values `1122` and `5566`, and find the minimum and maximum values using the `Long` class. (Hint: Use `Long.min(long, long)` and `Long.max(long, long)`).

```
public class Day3 {

        public static void main(String[] args) {

                long l1=112l;long l2=123l;

                long l=Long.max(l1,l2);

    System.out.println(l);

        }


}
```

k. Declare a long variable with the value `7`. Convert it to binary, octal, and hexadecimal strings using methods from the `Long` class. (Hint: Use `Long.toBinaryString(long)`, `Long.toOctalString(long)`, and `Long.toHexString(long)`).

```
public class Day3 {

        public static void main(String[] args) {

                long l1=112l;

                String l=Long.toBinaryString(l1);

                String l2=Long.toOctalString(l1);

                String l3=Long.toHexString(l1);

    System.out.println(l+" "+l2+" "+l3);

        }


}
```

l. Experiment with converting a `long` value into other primitive types or vice versa and observe the results.

6. Working with `java.lang.Float`

a. Explore the [Java API documentation for `java.lang.Float`] and observe its modifiers and super types.

b. Write a program to test how many bytes are used to represent a `float` value using the `BYTES` field. (Hint: Use `Float.BYTES`).

```
public class Day3 {

        public static void main(String[] args) {

    System.out.println(Float.BYTES);

        }


} //4
```

c. Write a program to find the minimum and maximum values of `float` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Float.MIN_VALUE` and `Float.MAX_VALUE`).

```
public class Day3 {

        public static void main(String[] args) {

    System.out.println(Float.MAX_VALUE+" "+Float.MIN_VALUE);

        }


}
```

```
3.4028235E38 1.4E-45
```

d. Declare a method-local variable `number` of type `float` with some value and convert it to a `String` using the `toString` method. (Hint: Use `Float.toString(float)`).

```
public class Day3 {

        public static void main(String[] args) {

                float f=123.3f;
```

```
                    String s1=Float.toString(f);

        System.out.println(s1);

            }



}
```

e. Declare a method-local variable `strNumber` of type `String` with some value and convert it to a `float` value using the `parseFloat` method. (Hint: Use `Float.parseFloat(String)`).

```
public class Day3 {



            public static void main(String[] args) {

//

          String s1 = "123";

          float i = Float.parseFloat(s1);

          System.out.println(i);

            }



}
```

f. Declare a method-local variable `strNumber` of type `String` with the value `"Ab12Cd3"` and attempt to convert it to a `float` value. (Hint: `parseFloat` method will throw a `NumberFormatException`).

it throws java.lang.NumberFormatException

g. Declare a method-local variable `number` of type `float` with some value and convert it to the corresponding wrapper class using `Float.valueOf()`. (Hint: Use `Float.valueOf(float)`).

```
public class Day3 {
```

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

//

 float number =123f;

 Float i = Float.valueOf(number);

        System.out.println(i);

            }



}
```

h. Declare a method-local variable `strNumber` of type `String` with some `float` value and convert it to the corresponding wrapper class using `Float.valueOf()`. (Hint: Use `Float.valueOf(String)`).

```
public class Day3 {

            public static void main(String[] args) {

                        String f="123.3f";

                        float s1=Float.valueOf(f);

     System.out.println(s1);

            }



}
```

i. Declare two float variables with values `112.3` and `984.5`, and add them using a method from the `Float` class. (Hint: Use `Float.sum(float, float)`).

```
public class Day3 {

            public static void main(String[] args) {

                        float f1=122.33f;float f2=77.77f;

                        float s1=Float.sum(f1,f2);

     System.out.println(s1);
```

```
          }


}
```

j. Declare two float variables with values `112.2` and `556.6`, and find the minimum and maximum values using the `Float` class. (Hint: Use `Float.min(float, float)` and `Float.max(float, float)`).

```
public class Day3 {

          public static void main(String[] args) {

                    float f1=122.33f;float f2=77.77f;

                    float s1=Float.max(f1,f2);

    System.out.println(s1);

          }


}
```

k. Declare a float variable with the value `-25.0f`. Find the square root of this value. (Hint: Use `Math.sqrt()` method).

```
public class Day3 {

          public static void main(String[] args) {

                    float f1=122.33f;

                    float s1=(float)Math.sqrt(f1);//returns double

    System.out.println(s1);

          }


}
```

l. Declare two float variables with the same value, `0.0f`, and divide them. (Hint: Observe the result and any special floating-point behavior).

```
public class Day3 {

        public static void main(String[] args) {

                float f1=0.0f;float f2=0.0f;

                float s1=f1/f2;;

    System.out.println(s1);

        }



}
```

it gives output as NaN or not a number for undefined divide operation result means it is handled at backend in the class itself

m. Experiment with converting a `float` value into other primitive types or vice versa and observe the results.

you can convert float to double or long fo others we need narrowing

7. Working with `java.lang.Double`

a. Explore the [Java API documentation for `java.lang.Double`] and observe its modifiers and super types.

b. Write a program to test how many bytes are used to represent a `double` value using the `BYTES` field. (Hint: Use `Double.BYTES`).

```
public class Day3 {

        public static void main(String[] args) {

    System.out.println(Double.BYTES);

        }



} //8
```

c. Write a program to find the minimum and maximum values of `double` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Double.MIN_VALUE` and `Double.MAX_VALUE`).

```
public class Day3 {

        public static void main(String[] args) {

    System.out.println(Double.MAX_VALUE+" "+Double.MIN_VALUE);

        }



}
```

```
1.7976931348623157E308 4.9E-324
```

d. Declare a method-local variable `number` of type `double` with some value and convert it to a `String` using the `toString` method. (Hint: Use `Double.toString(double)`).

```
public class Day3 {

        public static void main(String[] args) {

                double d=123.23;

                String s1=Double.toString(d);

    System.out.println(s1);

        }



}
```

e. Declare a method-local variable `strNumber` of type `String` with some value and convert it to a `double` value using the `parseDouble` method. (Hint: Use `Double.parseDouble(String)`).

```
public class Day3 {



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

```
//

        String s1 = "123";

        double i = Double.parseDouble(s1);

        System.out.println(i);

            }



}
```

f. Declare a method-local variable `strNumber` of type `String` with the value `"Ab12Cd3"` and attempt to convert it to a `double` value. (Hint: `parseDouble` method will throw a `NumberFormatException`).

it throws java.lang.NumberFormatException

g. Declare a method-local variable `number` of type `double` with some value and convert it to the corresponding wrapper class using `Double.valueOf()`. (Hint: Use `Double.valueOf(double)`).

```
public class Day3 {



            public static void main(String[] args) {

//

 double number =123f;

 Double i = Double.valueOf(number);

        System.out.println(i);

            }



}
```

h. Declare a method-local variable `strNumber` of type `String` with some `double` value and convert it to the corresponding wrapper class using `Double.valueOf()`. (Hint: Use `Double.valueOf(String)`).

```
public class Day3 {

        public static void main(String[] args) {

                String d="123.23";

                double s1=Double.valueOf(d);

    System.out.println(s1);

        }


}
```

i. Declare two double variables with values `112.3` and `984.5`, and add them using a method from the `Double` class. (Hint: Use `Double.sum(double, double)`).

```
public class Day3 {

        public static void main(String[] args) {

                double d1=12.32;double d2=56.65;

                double s1=Double.sum(d1,d2);

    System.out.println(s1);

        }


}
```

j. Declare two double variables with values `112.2` and `556.6`, and find the minimum and maximum values using the `Double` class. (Hint: Use `Double.min(double, double)` and `Double.max(double, double)`).

```
public class Day3 {

        public static void main(String[] args) {

                double d1=12.32;double d2=56.65;
                double s1=Double.max(d1,d2);

    System.out.println(s1);

        }


}
```

k. Declare a double variable with the value `-25.0`. Find the square root of this value. (Hint: Use `Math.sqrt()` method).

```
public class Day3 {

        public static void main(String[] args) {

                double d1=12.32;double d2=56.65;

                double s1=Math.sqrt(d1);

    System.out.println(s1);

        }


}
```

l. Declare two double variables with the same value, `0.0`, and divide them. (Hint: Observe the result and any special floating-point behavior).

```
public class Day3 {

        public static void main(String[] args) {

                double d1=0.0;double d2=0.0;

                double s1=d1/d2;

    System.out.println(s1);

        }
```

}

it gives output NaN

m. Experiment with converting a `double` value into other primitive types or vice versa and observe the results.

to convert double to int ,short , float we require type casting but we can convert int , float , short ,byte to double with explicit type casting

8. Conversion between Primitive Types and Strings

Initialize a variable of each primitive type with a user-defined value and convert it into `String`:

- First, use the `toString` method of the corresponding wrapper class. (e.g., `Integer.toString()`).
- Then, use the `valueOf` method of the `String` class. (e.g., `String.valueOf()`).

```
o public class Day3 {
o         public static void main(String[] args) {
o                 int i=1;float f=1.1f;double d=1.1;long l=1l;byte b=1;short s=1;
o                 String s1=Integer.toString(i);
o                 String s2=Float.toString(f);
o                 String s3=Double.toString(d);
o                 String s4=Long.toString(l);
o                 String s5=Byte.toString(b);
o                 String s6=Short.toString(s);
o     System.out.println(s1+" "+s2+" "+s3+" "+s4+" "+s5+" "+s6);
o         }
o
o }
o
```

9. Default Values of Primitive Types

Declare variables of each primitive type as fields of a class and check their default values. (Note: Default values depend on whether the variables are instance variables or static variables).

```
public class Day3 {

        int i ; float f;double d;char c;boolean b;



        public static void main(String[] args) {

                Day3 d1 = new Day3();

     System.out.println(d1.i+" "+d1.c+" "+d1.d+" "+d1.f);

        }



}
```

for instance variables default values are assigned when object is created while for static variables default values are assigned along with main method

10. Arithmetic Operations with Command Line Input

Write a program that accepts two integers and an arithmetic operator (+, -, *, /) from the command line. Perform the specified arithmetic operation based on the operator provided. (Hint: Use `switch-case` for operations).

```
package day3;

import java.util.Scanner;

public class Day3 {
        public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
   String c=sc.nextLine();
   switch(c) {
   case "*":int i = sc.nextInt();int j=sc.nextInt();
         System.out.println(i*j);
          break;
   case "+":int i1 = sc.nextInt();int j1=sc.nextInt();
```

```
                                        System.out.println(i1+j1);
                                        break;
    case "-":int i3 = sc.nextInt();int j3=sc.nextInt();
                                        System.out.println(i3-j3);
                                        break;
    case "/":int i2 = sc.nextInt();int j2=sc.nextInt();
                                        System.out.println(i2/j2);
                                                      break;
    }
           }

}
```