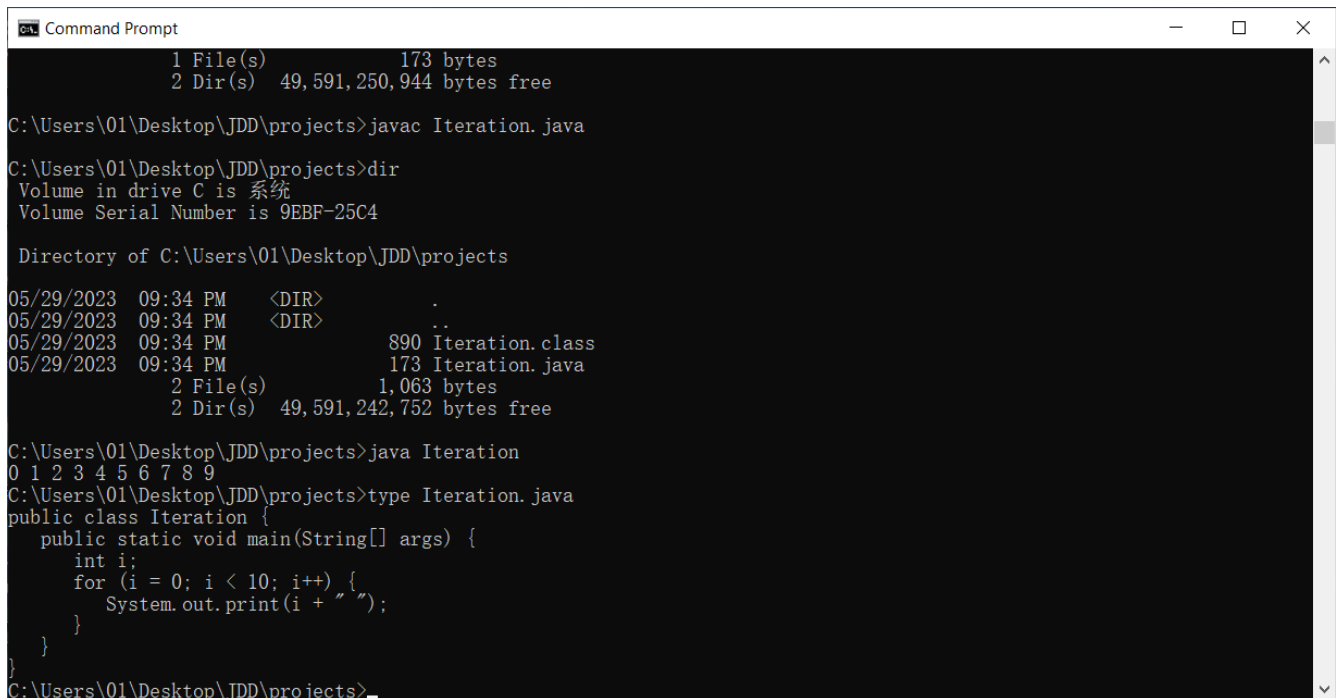


Iteration in Java - while loops and for loops

1. The *for* statement (aka the for loop)

A. In this example, the loop counter variable *i* is declared outside of the loop and initialized in the initialization statement of the loop. Since it is declared outside the loop, it is in scope after the loop exits, so its value can be printed to the console. (See fig. 2.)



```
Command Prompt
1 File(s)          173 bytes
2 Dir(s)  49,591,250,944 bytes free

C:\Users\01\Desktop\JDD\projects>javac Iteration.java

C:\Users\01\Desktop\JDD\projects>dir
Volume in drive C is 系统
Volume Serial Number is 9EBF-25C4

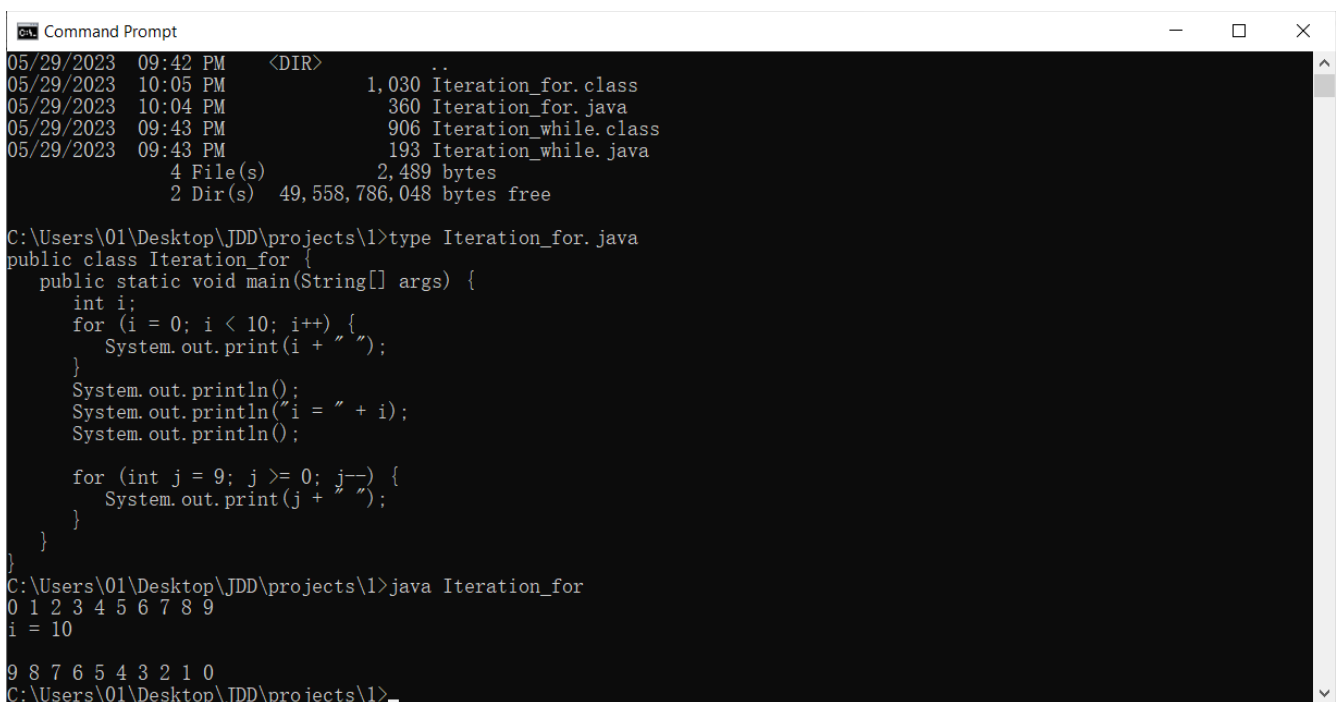
Directory of C:\Users\01\Desktop\JDD\projects

05/29/2023  09:34 PM  <DIR>          .
05/29/2023  09:34 PM  <DIR>          ..
05/29/2023  09:34 PM                890 Iteration.class
05/29/2023  09:34 PM                173 Iteration.java
                2 File(s)          1,063 bytes
                2 Dir(s)  49,591,242,752 bytes free

C:\Users\01\Desktop\JDD\projects>java Iteration
0 1 2 3 4 5 6 7 8 9
C:\Users\01\Desktop\JDD\projects>type Iteration.java
public class Iteration {
    public static void main(String[] args) {
        int i;
        for (i = 0; i < 10; i++) {
            System.out.print(i + " ");
        }
    }
}
C:\Users\01\Desktop\JDD\projects>
```

fig. 1

B. Here, the loop counter variable *j* is declared and initialized in the initialization statement of the loop.



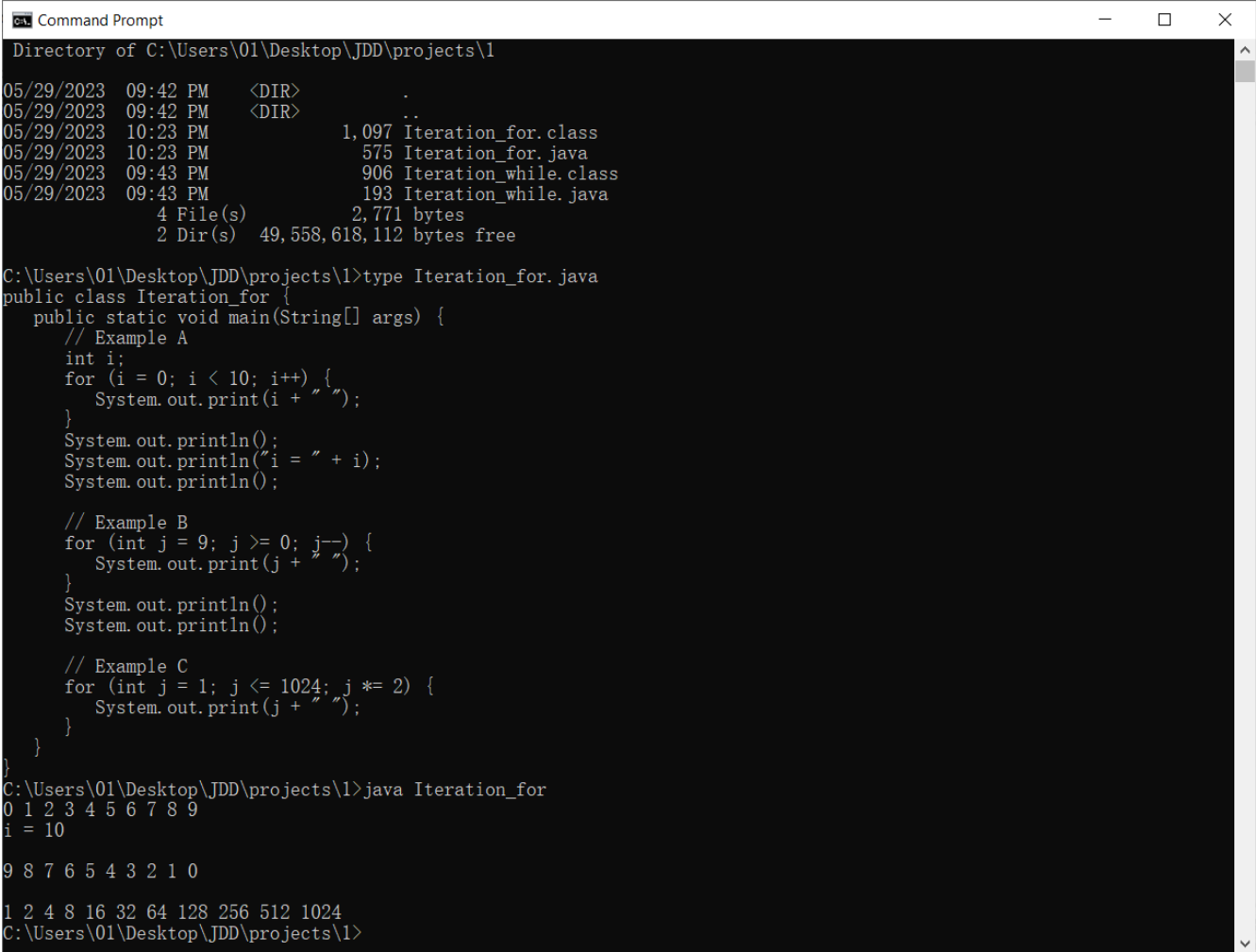
```
Command Prompt
05/29/2023  09:42 PM  <DIR>          ..
05/29/2023  10:05 PM                1,030 Iteration_for.class
05/29/2023  10:04 PM                360 Iteration_for.java
05/29/2023  09:43 PM                906 Iteration_while.class
05/29/2023  09:43 PM                193 Iteration_while.java
                4 File(s)          2,489 bytes
                2 Dir(s)  49,558,786,048 bytes free

C:\Users\01\Desktop\JDD\projects\1>type Iteration_for.java
public class Iteration_for {
    public static void main(String[] args) {
        int i;
        for (i = 0; i < 10; i++) {
            System.out.print(i + " ");
        }
        System.out.println();
        System.out.println("i = " + i);
        System.out.println();

        for (int j = 9; j >= 0; j--) {
            System.out.print(j + " ");
        }
    }
}
C:\Users\01\Desktop\JDD\projects\1>java Iteration_for
0 1 2 3 4 5 6 7 8 9
i = 10
9 8 7 6 5 4 3 2 1 0
C:\Users\01\Desktop\JDD\projects\1>
```

fig. 2

C. The update statement is not limited to increments or decrements. Any valid assignment statements can be used for the initialization and update.



```
Command Prompt
Directory of C:\Users\01\Desktop\JDD\projects\1
05/29/2023 09:42 PM <DIR> .
05/29/2023 09:42 PM <DIR> ..
05/29/2023 10:23 PM      1,097 Iteration_for.class
05/29/2023 10:23 PM      575 Iteration_for.java
05/29/2023 09:43 PM      906 Iteration_while.class
05/29/2023 09:43 PM      193 Iteration_while.java
          4 File(s)      2,771 bytes
          2 Dir(s) 49,558,618,112 bytes free

C:\Users\01\Desktop\JDD\projects\1>type Iteration_for.java
public class Iteration_for {
    public static void main(String[] args) {
        // Example A
        int i;
        for (i = 0; i < 10; i++) {
            System.out.print(i + " ");
        }
        System.out.println();
        System.out.println("i = " + i);
        System.out.println();

        // Example B
        for (int j = 9; j >= 0; j--) {
            System.out.print(j + " ");
        }
        System.out.println();
        System.out.println();

        // Example C
        for (int j = 1; j <= 1024; j *= 2) {
            System.out.print(j + " ");
        }
    }
}

C:\Users\01\Desktop\JDD\projects\1>java Iteration_for
0 1 2 3 4 5 6 7 8 9
i = 10

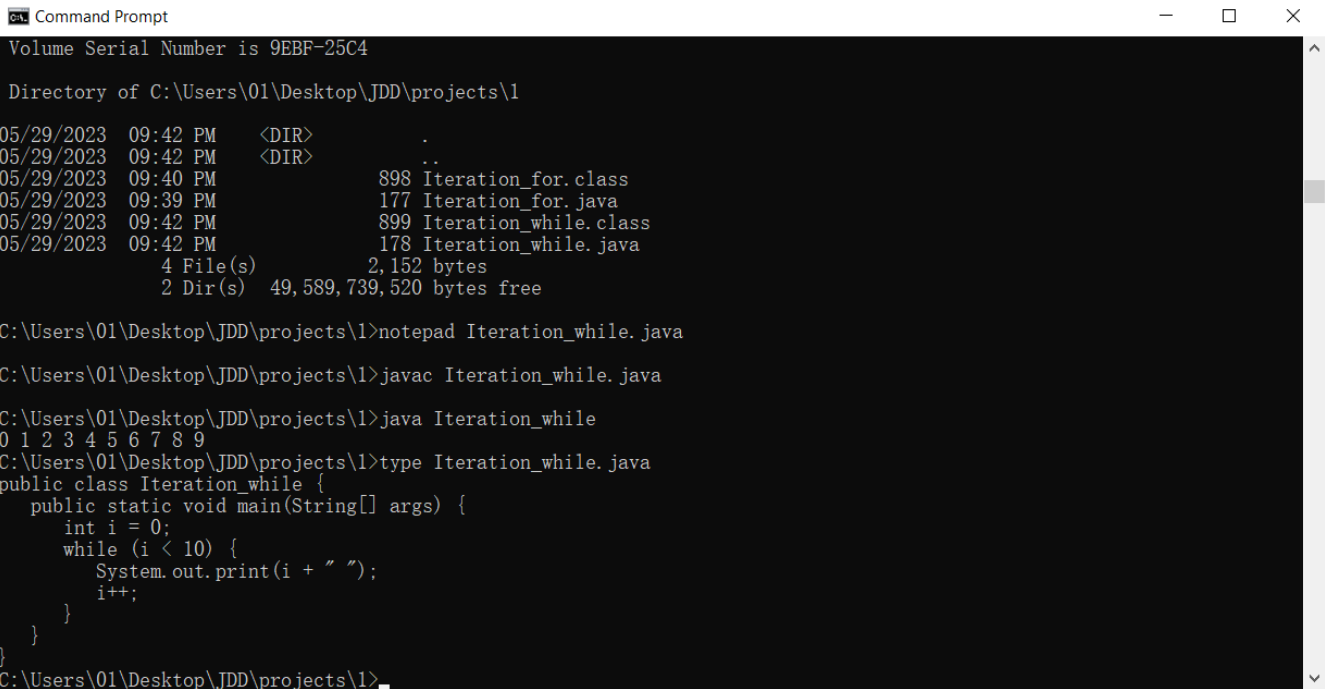
9 8 7 6 5 4 3 2 1 0

1 2 4 8 16 32 64 128 256 512 1024
C:\Users\01\Desktop\JDD\projects\1>
```

fig. 3

2. The *while* statement (aka the while loop)

A. The while loop in this example is equivalent to the for loop in the first example. To translate a for loop into a while loop, where should the update statement of the for loop be placed within the while loop?



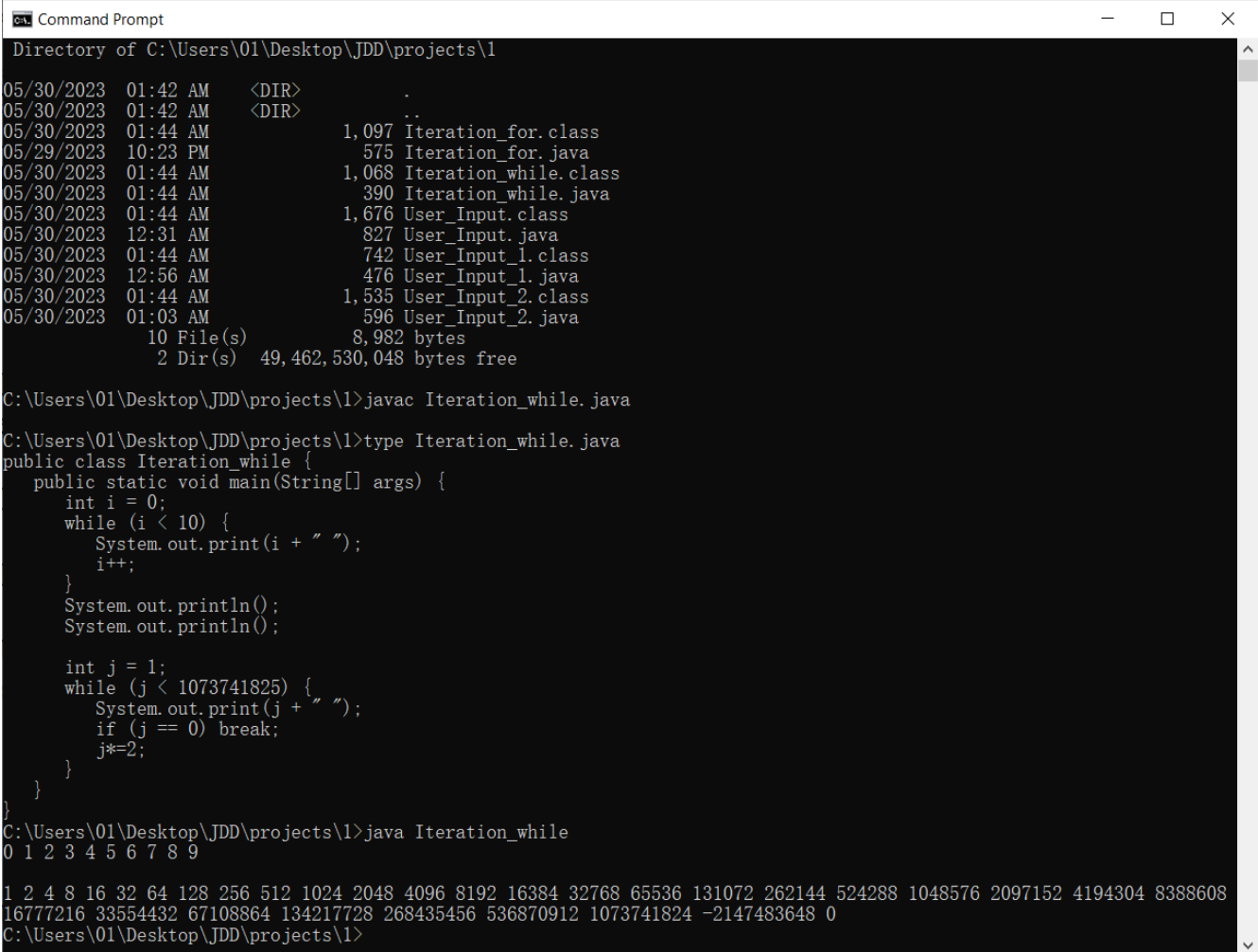
```
Command Prompt
Volume Serial Number is 9EBF-25C4

Directory of C:\Users\01\Desktop\JDD\projects\1
05/29/2023 09:42 PM <DIR>      .
05/29/2023 09:42 PM <DIR>      ..
05/29/2023 09:40 PM          898 Iteration_for.class
05/29/2023 09:39 PM          177 Iteration_for.java
05/29/2023 09:42 PM          899 Iteration_while.class
05/29/2023 09:42 PM          178 Iteration_while.java
           4 File(s)          2,152 bytes
           2 Dir(s) 49,589,739,520 bytes free

C:\Users\01\Desktop\JDD\projects\1>notepad Iteration_while.java
C:\Users\01\Desktop\JDD\projects\1>javac Iteration_while.java
C:\Users\01\Desktop\JDD\projects\1>java Iteration_while
0 1 2 3 4 5 6 7 8 9
C:\Users\01\Desktop\JDD\projects\1>type Iteration_while.java
public class Iteration_while {
    public static void main(String[] args) {
        int i = 0;
        while (i < 10) {
            System.out.print(i + " ");
            i++;
        }
    }
}
```

fig. 4

B. This example demonstrates the importance of being careful with loop conditions. Can you explain why the last two values of j are -2147483648 and 0 ? If the conditional break is removed, what will happen when the program runs? What loop condition would ensure the loop terminates?



```
Command Prompt
Directory of C:\Users\01\Desktop\JDD\projects\1
05/30/2023 01:42 AM <DIR> .
05/30/2023 01:42 AM <DIR> ..
05/30/2023 01:44 AM      1,097 Iteration_for.class
05/29/2023 10:23 PM      575 Iteration_for.java
05/30/2023 01:44 AM      1,068 Iteration_while.class
05/30/2023 01:44 AM      390 Iteration_while.java
05/30/2023 01:44 AM      1,676 User_Input.class
05/30/2023 12:31 AM      827 User_Input.java
05/30/2023 01:44 AM      742 User_Input_1.class
05/30/2023 12:56 AM      476 User_Input_1.java
05/30/2023 01:44 AM      1,535 User_Input_2.class
05/30/2023 01:03 AM      596 User_Input_2.java
10 File(s)            8,982 bytes
2 Dir(s) 49,462,530,048 bytes free

C:\Users\01\Desktop\JDD\projects\1>javac Iteration_while.java

C:\Users\01\Desktop\JDD\projects\1>type Iteration_while.java
public class Iteration_while {
    public static void main(String[] args) {
        int i = 0;
        while (i < 10) {
            System.out.print(i + " ");
            i++;
        }
        System.out.println();
        System.out.println();

        int j = 1;
        while (j < 1073741825) {
            System.out.print(j + " ");
            if (j == 0) break;
            j*=2;
        }
    }
}

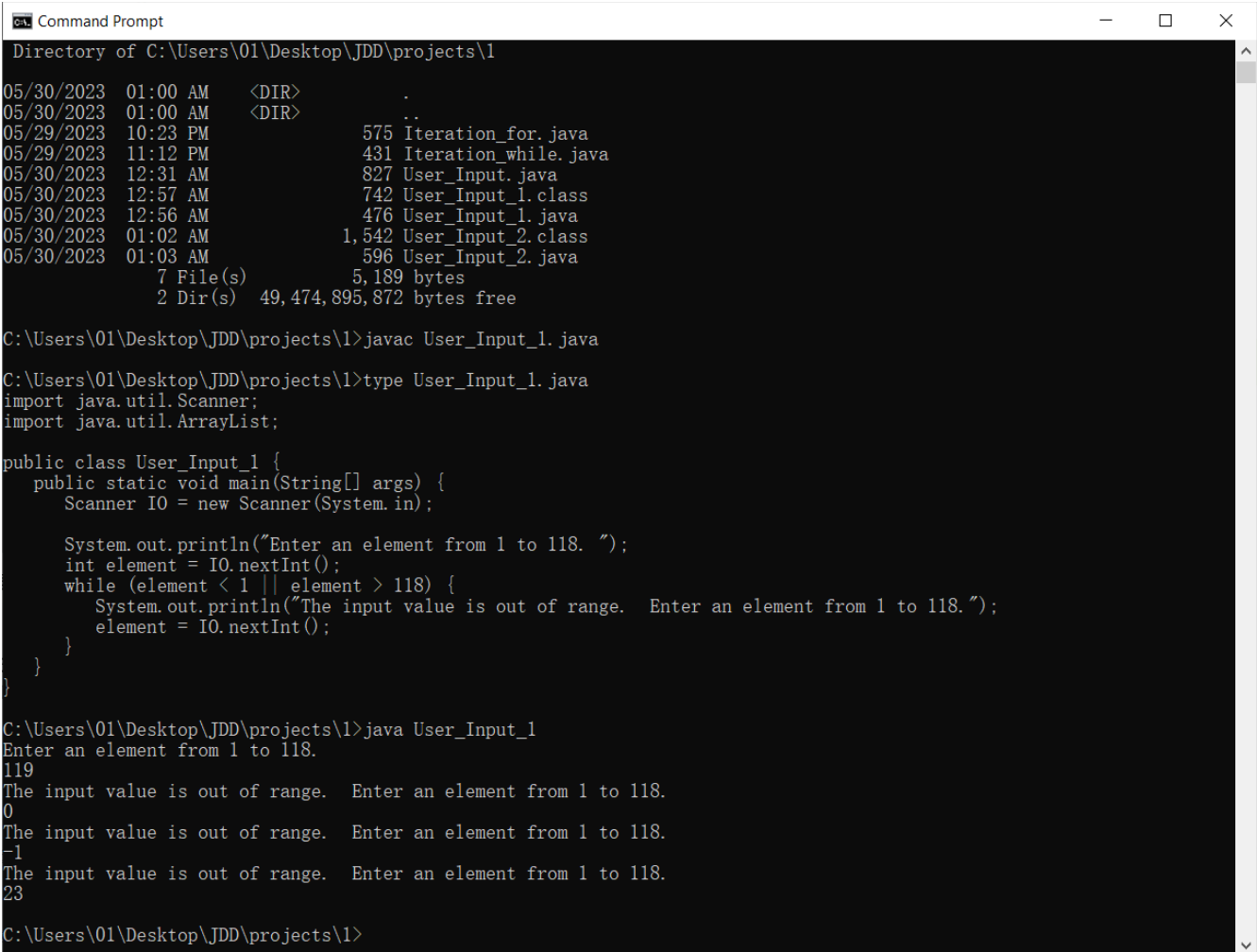
C:\Users\01\Desktop\JDD\projects\1>java Iteration_while
0 1 2 3 4 5 6 7 8 9
1 2 4 8 16 32 64 128 256 512 1024 2048 4096 8192 16384 32768 65536 131072 262144 524288 1048576 2097152 4194304 8388608
16777216 33554432 67108864 134217728 268435456 536870912 1073741824 -2147483648 0
C:\Users\01\Desktop\JDD\projects\1>
```

fig. 5

Comprehension check

1. Be able to explain the flow of control from the beginning to the end of a for loop, step by step, using the terms we learned in this lesson (initialization, condition, update, body).
2. Know how to translate a for loop into an equivalent while loop.
3. Be aware of loop conditions, to avoid infinite loops.
4. Understand the visibility of the for loop counter variable when it is declared before the loop and when it is declared in the initialization statement of the loop.

3. Using the while loop condition to filter out out-of-range user input values.



```
Command Prompt
Directory of C:\Users\01\Desktop\JDD\projects\1
05/30/2023 01:00 AM <DIR> .
05/30/2023 01:00 AM <DIR> ..
05/29/2023 10:23 PM          575 Iteration_for.java
05/29/2023 11:12 PM          431 Iteration_while.java
05/30/2023 12:31 AM          827 User_Input.java
05/30/2023 12:57 AM          742 User_Input_1.class
05/30/2023 12:56 AM          476 User_Input_1.java
05/30/2023 01:02 AM          1,542 User_Input_2.class
05/30/2023 01:03 AM          596 User_Input_2.java
          7 File(s)          5,189 bytes
          2 Dir(s) 49,474,895,872 bytes free

C:\Users\01\Desktop\JDD\projects\1>javac User_Input_1.java

C:\Users\01\Desktop\JDD\projects\1>type User_Input_1.java
import java.util.Scanner;
import java.util.ArrayList;

public class User_Input_1 {
    public static void main(String[] args) {
        Scanner IO = new Scanner(System.in);

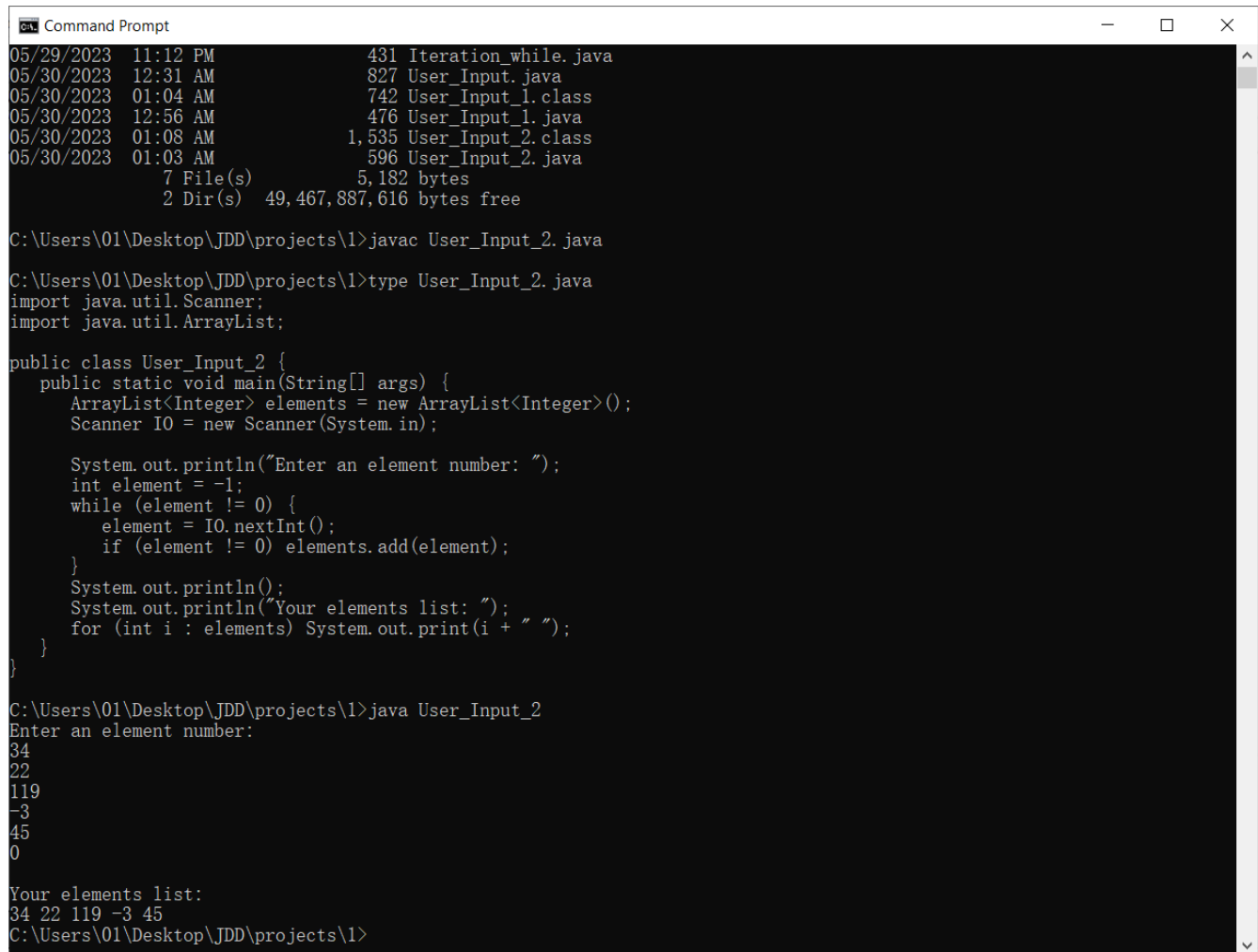
        System.out.println("Enter an element from 1 to 118. ");
        int element = IO.nextInt();
        while (element < 1 || element > 118) {
            System.out.println("The input value is out of range. Enter an element from 1 to 118.");
            element = IO.nextInt();
        }
    }
}

C:\Users\01\Desktop\JDD\projects\1>java User_Input_1
Enter an element from 1 to 118.
119
The input value is out of range. Enter an element from 1 to 118.
0
The input value is out of range. Enter an element from 1 to 118.
-1
The input value is out of range. Enter an element from 1 to 118.
23

C:\Users\01\Desktop\JDD\projects\1>
```

fig. 6

4. Using a while loop condition to detect the end of a list of user input data values. The user can input a reserved sentinel value, which is not a possible data value, to indicate the end of the list.



```
Command Prompt
05/29/2023 11:12 PM          431 Iteration_while.java
05/30/2023 12:31 AM          827 User_Input.java
05/30/2023 01:04 AM          742 User_Input_1.class
05/30/2023 12:56 AM          476 User_Input_1.java
05/30/2023 01:08 AM          1,535 User_Input_2.class
05/30/2023 01:03 AM          596 User_Input_2.java
      7 File(s)          5,182 bytes
      2 Dir(s) 49,467,887,616 bytes free

C:\Users\01\Desktop\JDD\projects\1>javac User_Input_2.java

C:\Users\01\Desktop\JDD\projects\1>type User_Input_2.java
import java.util.Scanner;
import java.util.ArrayList;

public class User_Input_2 {
    public static void main(String[] args) {
        ArrayList<Integer> elements = new ArrayList<Integer>();
        Scanner IO = new Scanner(System.in);

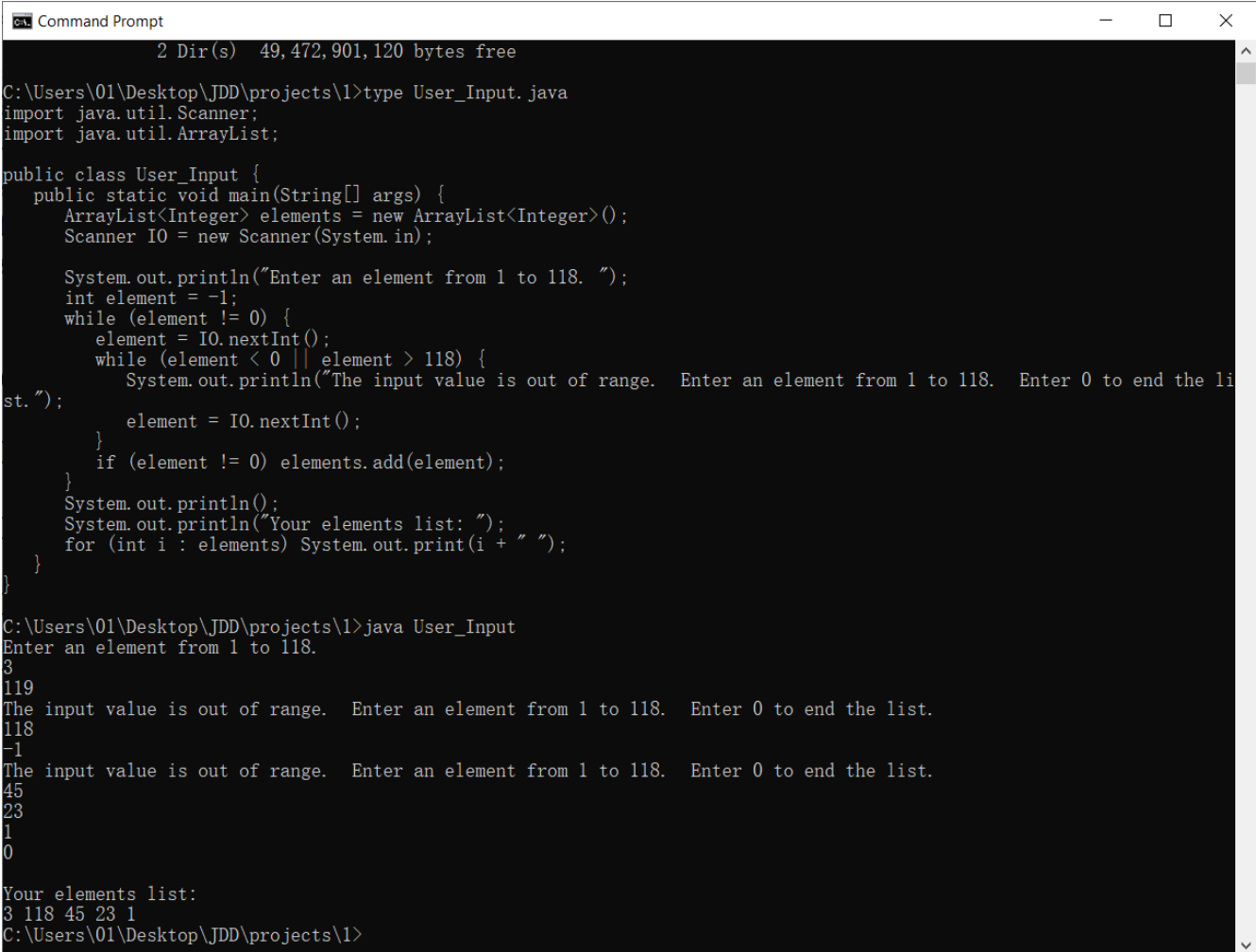
        System.out.println("Enter an element number: ");
        int element = -1;
        while (element != 0) {
            element = IO.nextInt();
            if (element != 0) elements.add(element);
        }
        System.out.println();
        System.out.println("Your elements list: ");
        for (int i : elements) System.out.print(i + " ");
    }
}

C:\Users\01\Desktop\JDD\projects\1>java User_Input_2
Enter an element number:
34
22
119
-3
45
0

Your elements list:
34 22 119 -3 45
C:\Users\01\Desktop\JDD\projects\1>
```

fig. 7

5. This example uses nested while loops to combine the previous two examples (of sentinels and input filtering) into a single program that allows the user to input a list of data terminated by a sentinel value, from which values that are out of range are filtered out. A for-each loop is used to display the list.



```
Command Prompt
2 Dir(s) 49,472,901,120 bytes free

C:\Users\01\Desktop\JDD\projects\1>type User_Input.java
import java.util.Scanner;
import java.util.ArrayList;

public class User_Input {
    public static void main(String[] args) {
        ArrayList<Integer> elements = new ArrayList<Integer>();
        Scanner IO = new Scanner(System.in);

        System.out.println("Enter an element from 1 to 118. ");
        int element = -1;
        while (element != 0) {
            element = IO.nextInt();
            while (element < 0 || element > 118) {
                System.out.println("The input value is out of range. Enter an element from 1 to 118. Enter 0 to end the list.");
            }
            element = IO.nextInt();
            if (element != 0) elements.add(element);
        }
        System.out.println();
        System.out.println("Your elements list: ");
        for (int i : elements) System.out.print(i + " ");
    }
}

C:\Users\01\Desktop\JDD\projects\1>java User_Input
Enter an element from 1 to 118.
3
119
The input value is out of range. Enter an element from 1 to 118. Enter 0 to end the list.
118
-1
The input value is out of range. Enter an element from 1 to 118. Enter 0 to end the list.
45
23
1
0

Your elements list:
3 118 45 23 1
C:\Users\01\Desktop\JDD\projects\1>
```

fig. 8

Lab Exercise

Write a program that chooses a number between 1 and 1,000,000 and asks the user to guess the number. If the guess is correct, the program should terminate. If the guess is incorrect, the program should ask the user for another guess. At the end, display the number of guesses the user took to get the right answer.

For extra credit, determine an upper bound on the number of guesses needed to find the number. Compare the number of guesses the user took to find the number with this upper bound to determine a score.

See fig. 8 for Scanner code and loop structures that you might find helpful.