CS1101 Programming and Problem Solving

Dr. Gina Bai

Spring 2023

Logistics

- **ZY-5A** on **zyBook > Assignments**
  - Due: **Wednesday, March 1**, at 11:59pm
- **PA06 – W, A, B** on **zyBook > Chap 11**
  - Due: **Thursday, March 2**, at 11:59pm
- Midterm Exam 1 regrade requests due Tuesday, Feb 28

Scanner hasNext Methods

Java Scanner API

Recap – next Methods

Return	Method	Description
String	next()	Finds and returns the next complete token from this scanner.
String	nextLine()	Advances this scanner past the current line and returns the input that was skipped.
int	nextInt()	Scans the next token of the input as an int.
double	nextDouble()	Scans the next token of the input as a double

Error Handling

- `InputMismatchException`
  - If the next token does not match the pattern for the expected type, or is out of range for the expected type

```java
import java.util.Scanner;

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

        Scanner input = new Scanner(System.in);

        System.out.print("Enter place (int): ");
        int place = input.nextInt();

        if (place <= 3) {
            System.out.println("You earned a medal!");
        } else {
            System.out.println("Finisher!");
        }
    }
}
```

```
$ javac CheckRaceResults.java
$ java CheckRaceResults
Enter place (int): 1
You earned a medal!

$ java CheckRaceResults
Enter place (int): one
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:939)
        at java.base/java.util.Scanner.next(Scanner.java:1594)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
        at CheckRaceResults.main(CheckRaceResults.java:9)
```

Error Handling

- `InputMismatchException`
  - If the next token does not match the pattern for the expected type, or is out of range for the expected type
- `NoSuchElementException`
  - If the input is exhausted

hasNext Methods

next methods – Actual READ in
hasNext methods – CHECK only

Return	Method	Description
String	next()	Finds and returns the next complete token from this scanner.
boolean	**hasNext()**	**Returns true if this scanner has another token in its input.**
String	nextLine()	Advances this scanner past the current line and returns the input that was skipped.
boolean	**hasNextLine()**	**Returns true if there is another line in the input of this scanner.**
int	nextInt()	Scans the next token of the input as an int.
boolean	**hasNextInt()**	**Returns true if the next token in this scanner's input can be interpreted as an int value using the nextInt() method.**
double	nextDouble()	Scans the next token of the input as a double
boolean	**hasNextDouble()**	**Returns true if the next token in this scanner's input can be interpreted as a double value using the nextDouble() method.**

Q: What's the output given the input as ➡️

```
$ javac HasNextMethods.java
$ java HasNextMethods
Token(s): CS1101
hasNextInt = false
hasNextDouble = false
hasNext = true
hasNextLine = true

$ java HasNextMethods
Token(s): CS 1101
hasNextInt = false
hasNextDouble = false
hasNext = true
hasNextLine = true
```

```
$ java HasNextMethods
Token(s): 1101
hasNextInt = true
hasNextDouble = true
hasNext = true
hasNextLine = true

$ java HasNextMethods
Token(s): 110.1
hasNextInt = false
hasNextDouble = true
hasNext = true
hasNextLine = true
```

```java
import java.util.Scanner;

public class HasNextMethods {
    public static void main (String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Token(s): ");

        System.out.println("hasNextInt = " + input.hasNextInt());
        System.out.println("hasNextDouble = " + input.hasNextDouble());
        System.out.println("hasNext = " + input.hasNext());
        System.out.println("hasNextLine = " + input.hasNextLine());
    }
}
```

hasNext methods DO NOT consume input

Robust Programs!

- Robustness
  - The degree to which erroneous situations are handled gracefully
- Want to write programs that execute when we present illegal data
  - Testing provides the illegal data
  - Now want to handle it

Handling User Errors

```java
import java.util.Scanner;

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

        Scanner input = new Scanner(System.in);

        System.out.print("Enter place (int): ");

        // Check if next token can be read in as an int
        while (!input.hasNextInt()) {
            // discard the invalid token by reading in the token
               but not assigning it to any variable
            input.next();
            System.out.println("Not an int, try again.");
            System.out.print("Enter an int: ");
        }
        // ASSERT: the next token can be read as an int
        int place = input.nextInt();

        if (place <= 3) {
            System.out.println("You earned a medal!");
        } else {
            System.out.println("Finisher!");
        }
    }
}
```

```
$ javac CheckRaceResults.java
$ java CheckRaceResults
Enter place (int): one
Not an int, try again.
Enter an int: 1one
Not an int, try again.
Enter an int: 1
You earned a medal!
```

TopHat

Q: What is the output from the following code segment?

```
int sum = 0, number;
for (number = 1; number <= 4; ++number); {
        sum = sum + number;
}
System.out.println(sum);
```

Output: 5

TopHat

Q: What does the following method return when passed the string "cab" where the variable c contains the character 'c'?

```
public static int indexOfChar (String s, char c) {
        for (int i = 0; i < s.length(); ++i) {
                if (s.charAt(i) == c) {
                        return i;
                }
        }
}
```

Doesn't compile
Missing a return statement

Pseudocode

A description of an algorithm using a mixture of English and Java.

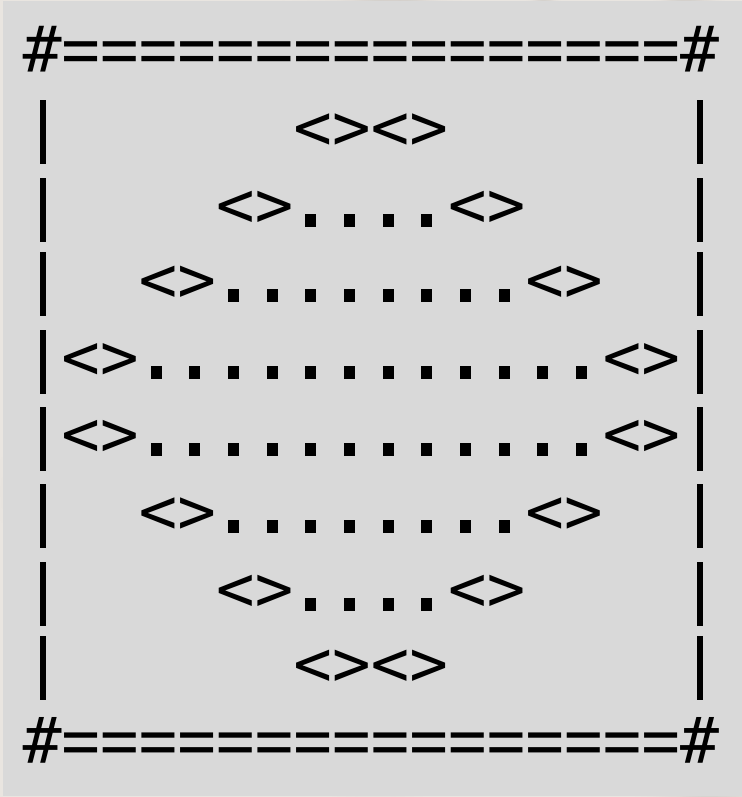
For example:

```
print 12 stars
for (each of the five lines)
    print a star
    print 10 spaces
    print a star
print 12 stars
```

```
************
*          *
*          *
*          *
*          *
*          *
************
```

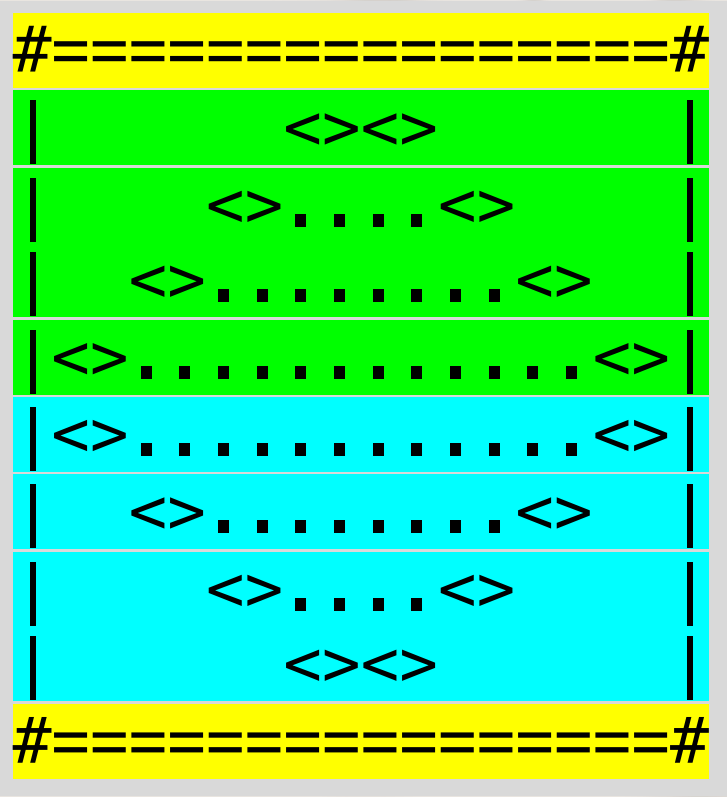
Brainstorm: How to Reproduce →

Q: How to decompose the pattern?



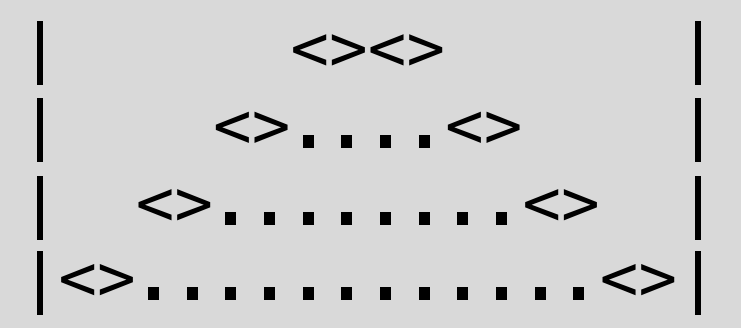
Brainstorm: How to Reproduce →

1. Print a line with a # sign, 16 = signs, and another # sign
2. Print the top half, for each line
    1. Print one | symbol
    2. Print several spaces (decreasing, 6 → 4 → 2 → 0 spaces)
    3. Print one pair of <> symbols
    4. Print dots (increasing, 0 → 4 → 8 → 12 dots)
    5. Print one pair of <> symbols
    6. Print several spaces (decreasing, 6 → 4 → 2 → 0 spaces)
    7. Print one | symbol
3. Print the bottom half (the upside-down version of the top half)
4. Print a line with a # sign, 16 = signs, and another # sign, again



Print the top half, for each line

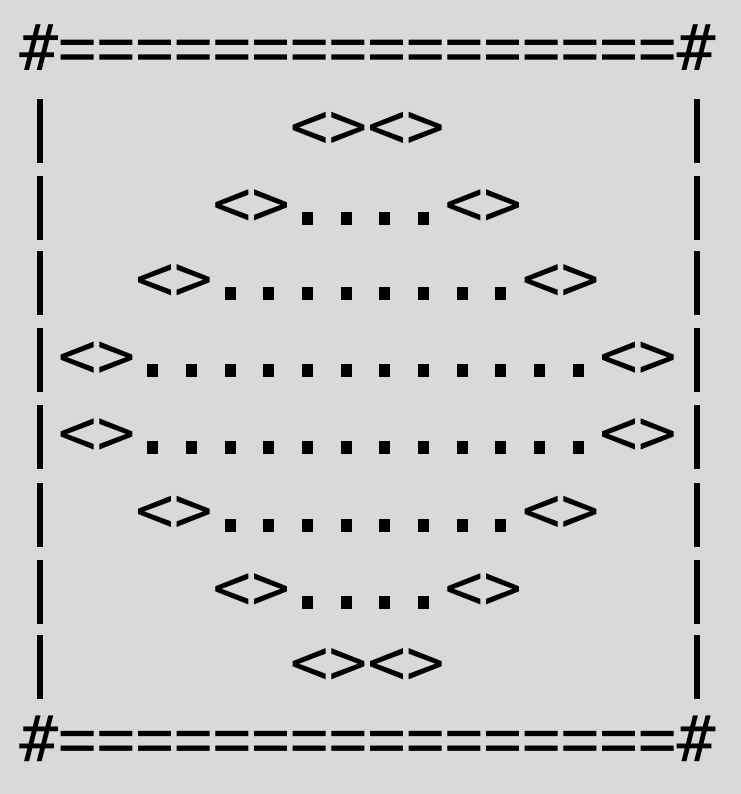
1. Print one | symbol
2. Print several spaces (decreasing, 6 → 4 → 2 → 0 spaces)
3. Print one pair of <> symbols
4. Print dots (increasing, 0 → 4 → 8 → 12 dots)
5. Print one pair of <> symbols
6. Print several spaces (decreasing, 6 → 4 → 2 → 0 spaces)
7. Print one | symbol



Line (i)	#spaces (2 * (4 - i))	#dots (4 * (i - 1))
1	6	0
2	4	4
3	2	8
4	0	12

Coding Practice – Part 1

Generate the pattern →



Sample Solution – Part 1

```
public class ComplexPattern {
    public static void main(String[] args) {
        line();
        topHalf();
        bottomHalf();
        line();
    }

    public static void topHalf() {
        for (int line = 1; line <= 4; ++line) {
            System.out.print("|");

            for (int space = 0; space < 2 * (4 - line); ++space) {
                System.out.print(" ");
            }

            System.out.print("<>");

            for (int dot = 0; dot < 4 * (line - 1); ++dot) {
                System.out.print(".");
            }

            System.out.print("<>");

            for (int space = 0; space < 2 * (4 - line); ++space) {
                System.out.print(" ");
            }

            System.out.println("|");
        }
    }

    public static void bottomHalf() {
        for (int line = 4; line >= 1; --line) {
            System.out.print("|");

            for (int space = 0; space < 2 * (4 - line); ++space) {
                System.out.print(" ");
            }

            System.out.print("<>");

            for (int dot = 0; dot < 4 * (line - 1); ++dot) {
                System.out.print(".");
            }

            System.out.print("<>");

            for (int space = 0; space < 2 * (4 - line); ++space) {
                System.out.print(" ");
            }

            System.out.println("|");
        }
    }

    public static void line() {
        System.out.println("#===============#");
    }
}
```

Coding Practice – Part 2

Modify the program, so it

- Prompts the user for the size
- Reprompts the user if the input is not an int
- Resize the pattern given the user input

```
$ java ResizableComplexPattern
Enter pattern size (int): four
Not an int, try again: 4.0
Not an int, try again: 4
#===============#
|       <><>       |
|     <>....<>     |
|   <>........<>   |
| <>............<> |
| <>............<> |
|   <>........<>   |
|     <>....<>     |
|       <><>       |
#===============#

$ java ResizableComplexPattern
Enter pattern size (int): 3
#============#
|     <><>     |
|   <>....<>   |
| <>........<> |
| <>........<> |
|   <>....<>   |
|     <><>     |
#============#
```

Sample Solution – Part 2

```java
import java.util.Scanner;

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

        Scanner input = new Scanner(System.in);

        System.out.print("Enter pattern size (int): ");
        // Checks if the next token can be read as an int
        while (!input.hasNextInt()) {
            input.next(); // Discard the invalid token
            System.out.print("Not an int, try again: ");
        }
        // ASSERT: the next token can be read as an int
        int size = input.nextInt();

        line(size);
        topHalf(size);
        bottomHalf(size);
        line(size);
    }

    public static void line(int size) {
        System.out.print("#");
        for (int i = 0; i < size; ++i){
            System.out.print("====");
        }
        System.out.println("#");
    }

    public static void topHalf(int size) {
        for (int line = 1; line <= size; ++line) {
            System.out.print("|");

            for (int space = 0; space < 2 * (size - line); ++space) {
                System.out.print(" ");
            }

            System.out.print("<>");

            for (int dot = 0; dot < 4 * (line - 1); ++dot) {
                System.out.print(".");
            }

            System.out.print("<>");

            for (int space = 0; space < 2 * (size - line); ++space) {
                System.out.print(" ");
            }

            System.out.println("|");
        }
    }

    public static void bottomHalf(int size) {
        for (int line = size; line >= 1; --line) {
            // Same as the code in topHalf
        }
    }
}
```