CS1101 Programming and Problem Solving

Dr. Gina Bai

Spring 2023

Logistics

TWO free late days are added to your “account”

- Up to two late days can be used per assignment
  - One late day extends a PA (all subproblems) deadline by 24 hours

- The unused late days will be rolled over
  - Four in total throughout the semester

- Cannot be used on past due assignments

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

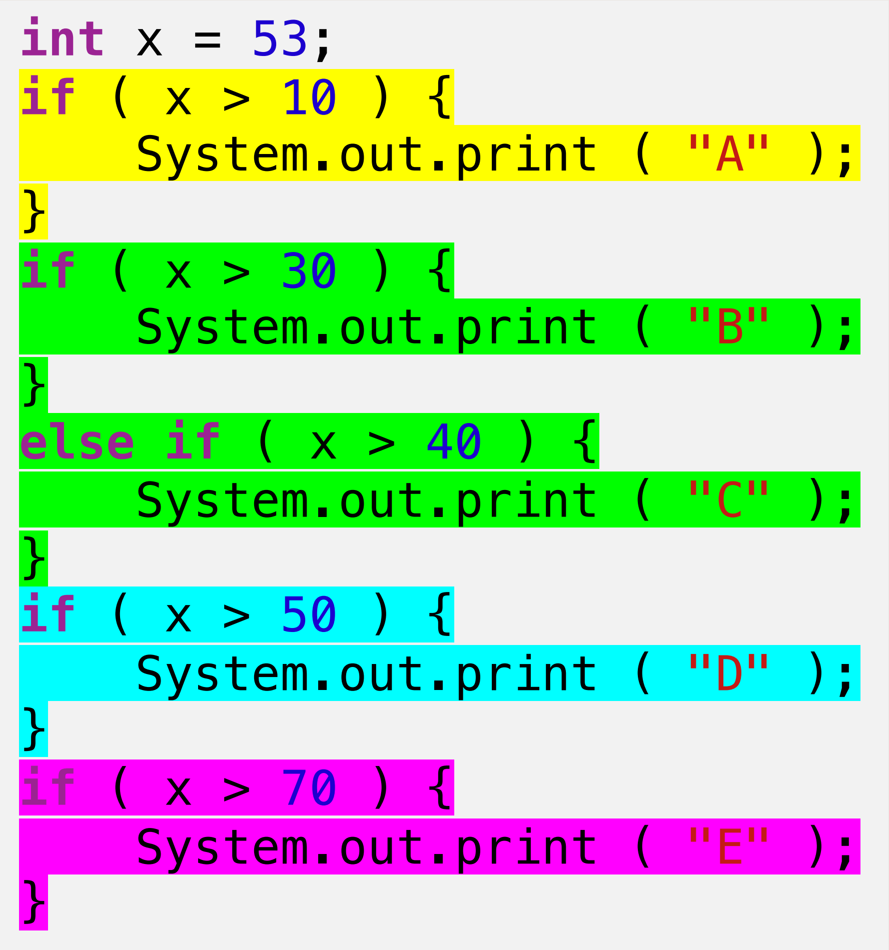
Recap – if statements

```java
int x = 53;
if ( x > 10 ) {
    System.out.print ( "A" );
}
if ( x > 30 ) {
    System.out.print ( "B" );
}
else if ( x > 40 ) {
    System.out.print ( "C" );
}
if ( x > 50 ) {
    System.out.print ( "D" );
}
if ( x > 70 ) {
    System.out.print ( "E" );
}
```

Q: What is output after the code executes?

1) A
2) D
3) ABCDE
4) ABCD
5) ABD

Recap – if statements



```java
int x = 53;
if ( x > 10 ) {
    System.out.print ( "A" );
}
if ( x > 30 ) {
    System.out.print ( "B" );
}
else if ( x > 40 ) {
    System.out.print ( "C" );
}
if ( x > 50 ) {
    System.out.print ( "D" );
}
if ( x > 70 ) {
    System.out.print ( "E" );
}
```

Q: What is output after the code executes?

1) A
2) D
3) ABCDE
4) ABCD
[✓ 5) ABD]

Recap – loops

Consider the following abstraction of a for loop where <1>, <2>, <3>, and <4> represents legal code in the indicated locations:

```
for ( <1>; <2>; <3> ) {
        <4>;
}
```

Q: Which of the following while loops has the same functionality as the above for loop?

A

```
<1>;
<3>;
while ( <2> ) {
     <4>;
     <3>;
}
```

B

```
<1> ;
while ( <2> ) {
     <3>;
     <4>;
}
```

C ✓

```
<1>;
while ( <2> ) {
     <4>;
     <3>;
}
```

D

```
<1>;
while ( !<2> ) {
     <4>;
     <3>;
}
```

Challenge

Doesn't compile.
Error: char cannot be dereferenced
Correction: `Character.isUpperCase(userString.charAt(i))`

Q: What will the code segment output?

```
Scanner console = new Scanner(System.in);
System.out.print("Enter the string: ");
String userString = console.nextLine();
int i = 0;
while (userString.charAt(i).isUpperCase() == false) {
      ++i;
}
System.out.println("First capital is position " + i);
```

Challenge

Doesn't run.
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range

Q: What will the code segment output?

```
Scanner console = new Scanner(System.in);
System.out.print("Enter the string: ");
String userString = console.nextLine();
int i = 0;
while (Character.isUpperCase(userString.charAt(i)) == false) {
      ++i;
}
System.out.println("First capital is position " + i);
```

E.g., an input String with all lowercase letters cannot break the while loop, such as "sss". The while loop will check the charAt(3)

Random Numbers

zyBook Chap 5.12

When to use random numbers?

- Games
  - Typing games
  - Shuffle cards, roll dice…
  - Flashcards
- Statistical sampling
- Cryptography

Pseudo-random

- **Pseudo-random**:
  - Numbers that, although they are derived from predictable and well-defined algorithms, mimic the properties of numbers chosen at random

- The pseudo-random number generator generates a number based on a **seed**, which is the **current time**, which is different for each program run

Random Number in Java

1. **Math.random()** method

2. **Random** object
   - `import java.util.Random;`

Math.random()

- Returns a random number between **[0.0, 1.0)**
- Can use multiplication to extend the range
- Example:

```
double random = Math.random();       // [0.0, 1.0)
double random = 2.0 * Math.random(); // [0.0, 2.0)
```

Random Objects

- Must `import java.util.Random`
- Construct it with the keyword new

```
Random rand = new Random();
```

Return	Method	Description	Example
int	**nextInt()**	**Random int between -2^{31} and (2^{31} - 1)**	**int x = rand.nextInt();**
int	**nextInt(max)**	**Random int between [0, (max - 1)]**	**int y = rand.nextInt(10);**
double	nextDouble()	Random real # between [0.0, 1.0)	double z = rand.nextDouble();
boolean	nextBoolean()	Random logical value of true or false	boolean b = rand.nextBoolean();

Q: What is the range of the result of integers a, b, c, and d?
Be specific with inclusive vs. exclusive.

```
Random rand = new Random();
```

`int a = rand.nextInt(50);`	**[0, 49]**
`int b = rand.nextInt(5)+10;`	**[10, 14]**
`int c = rand.nextInt(10)+5;`	**[5, 14]**
`int d = rand.nextInt(50)–25;`	**[-25, 24]**

nextInt(max)
Returns a random int between **[0, (max - 1)]**

Q: What's wrong with the following code?

```java
import java.util.Random;

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

        Random r = new Random();

        System.out.println("My random value is: " + (r.nextInt(101)));
        System.out.println("My random value plus 1: " + (r.nextInt(101) + 1));
        System.out.println("My random value times 5: " + (r.nextInt(101) * 5));
    }
}
```

```
$ java RandomSingleValue
My random value is: 65
My random value plus 1: 2
My random value times 5: 275

$ java RandomSingleValue
My random value is: 4
My random value plus 1: 97
My random value times 5: 375

$ java RandomSingleValue
My random value is: 31
My random value plus 1: 32
My random value times 5: 165
```

The `nextInt(101)` method generates a random integer between [0, 100] every time it's called.

Corrected

```
import java.util.Random;

public class RandomSingleValue {

    public static void main (String[] args) {

        Random r = new Random();
        int val = r.nextInt(101);

        System.out.println("My random value is: " + val);
        System.out.println("My random value plus 1: " + (val + 1));
        System.out.println("My random value times 5: " + (val * 5));
    }
}
```

Generate only one random integer between [0, 100]

```
$ java RandomSingleValue
My random value is: 78
My random value plus 1: 79
My random value times 5: 390

$ java RandomSingleValue
My random value is: 60
My random value plus 1: 61
My random value times 5: 300

$ java RandomSingleValue
My random value is: 2
My random value plus 1: 3
My random value times 5: 10
```

Unit Testing

zyBook Chap 5.11

Testing

- “The ***dynamic*** verification of the behavior of a program on a finite set of test cases, ***suitably selected*** from the usually infinite executions domain, against the ***expected behavior***”

[ISO/IEC TR 19759:2005. Software Engineering – Guide to the Software Engineering Body of Knowledge (SWEBOK)]

- A process of **verifying the behavior** of our programs and **revealing software faults** (i.e., logic errors)

Unit Testing

- The most basic level of software testing
- Testing the functionality of **individual methods**
  - Independent paths within the source code
  - Logical decisions as both true and false
  - Loops at their boundaries
  - Internal data structures
  - …

Testing Strategies

- **Test Requirements**
- **Test Equivalence Classes**
- **Test Boundary Values**
- Test All Paths
- Test Exceptions

Testing Strategies – Test Requirements

- Testing the main functionality of the method

For example, to test the method

```
      public static boolean isPalindrome (int userNum)
```

in PA05-A: Numeric Palindromes.

We need to verify if this method can return `true` when the parameter `userNum` is a palindrome, and return `false` otherwise.

Testing Strategies – Test Equivalence Classes

- Testing representative values from equivalence classes
  - We break up the possible inputs for each parameter into equivalence classes and test representative values for each parameter
  - Preferably the “middle” input values
- Ensure each equivalence class is tested

Testing Strategies – Test Equivalence Classes

As stated in the Program Description, the `userNum` should be within the range of 1 – 999, inclusive.

Hence, we divide the range of possible inputs into

- < 1
- 1 - 999
- > 999

Can the possible user input be further divided into more equivalent classes?

Testing Strategies – Test Equivalence Classes

As stated in the Program Description, the `userNum` should be within the range of 1 – 999, inclusive.

Hence, we divide the range of possible inputs into

- < 1
- 1 – 999
  - 1 – 9
  - 10 – 99
  - 100 – 999
- > 999

Testing Scenarios/Values:
Test our program with at least one value (in this case, a palindrome and a non-palindrome) from each equivalence class.

Testing Strategies – Test Boundary Values

- Once representative values of a method are tested, boundary values (if any) between the equivalence classes should be tested

- < 1
- 1 – 999
  - 1 – 9
  - 10 – 99
  - 100 – 999
- > 999

Boundary Values (in this case):
-1, **0**, 1, 9, 10, 99, 100, 999, 1000

How to improve the PA05-A test cases?

Current PA05-A test cases verify the program behavior given:

- -15
- 151
- 511
- 999
- 1000
- 456