



Day 12: Midterm Review





Agenda

- Quiz 5
- Variables
- Primitive Data Types
- Casting
- Random
- Binary
- Conditional Statements
- While Loops
- For Loops
- Strings
- Arrays
- 2D Arrays
- Static Functions





Quiz 5



Variables



```
data type variable name = value;
```

Store a value into a variable with a specific data type.

```
int x = 5;
```

```
String name = "Edward";
```



Primitive Data Types



- byte
- short
- int
- long
- char
- boolean
- float
- double



byte

Values between -2^7 to $2^7 - 1$

```
byte a = 127;
```




short

Values between -2^{15} to $2^{15} - 1$

```
short a = 10830;
```



int

Values between -2^{31} to $2^{31} - 1$

```
int a = 1038147;
```



long

Values between -2^{63} to $2^{63} - 1$

Require an L or l at the end.

```
long a = 391038147L;
```



char

Unicode 16-bit single character

```
char e = 'i';
```



boolean

True/False value

```
boolean b = true;
```



float

Decimal value, up to 7 decimals
require f at the end.

```
float f = 1.0f;
```



double

Decimal value, up to 15 decimals

```
double d = 1.35;
```



Casting



Casting

Converting between one primitive to another
primitive

```
float a = 64.0138;
```

```
int f = (int) a;
```



Random Numbers



Random numbers

```
(int) ((Math.random() * (max - min) + min));
```



Random numbers (3-25)

```
(int) ((Math.random() * (25 - 3) + 3));
```



Binary



Decimal to Binary

$n : 53$ Method 1

① Powers of 2 less than n

2^n	n
1	0
2	1
4	2
8	3
16	4
32	5

③ Start subtracting powers of 2 from num. If power is more than difference, mark 0 then continue. If power of 2 subtracted, write the 1.

$$1 \times 2^5 + 1 \times 2^4 + \underbrace{0 \times 2^3}_{\textcircled{1}} + 1 \times 2^2 + \underbrace{0 \times 2^1}_{\textcircled{2}} + \underbrace{1 \times 2^0}_{\textcircled{2}}$$

110101

④ Given we need 8-bits, add 0s in front until 8 bits.

00110101

$$\begin{array}{r} 53 \\ - 32 \\ \hline 21 \\ - 16 \\ \hline 5 \\ - 4 \\ \hline 1 \\ - 1 \\ \hline 0 \end{array}$$

$\textcircled{1} 5 < 8 \rightarrow$

$\textcircled{2} 1 < 2 \rightarrow$



Alternative Approach

Method 2

 $n = 53$

① Table of powers of 2 up to 2^7 .

2^n	n
1	0
2	1
4	2
8	3
16	4
32	5
64	6
128	7

③ If $n > 0$, Start with 0000 0000

③a Subtract the nearest power of 2 from n .
Place a 1 in the binary based on the power subtracted

$-2^3 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$

0 0 1 1 0 1 0 1

$$\begin{array}{r} 53 \\ - 32 \leftarrow 2^5 \\ \hline 21 \\ - 16 \leftarrow 2^4 \\ \hline 5 \\ 4 \leftarrow 2^2 \\ \hline 1 \leftarrow 2^0 \end{array}$$

④ if $n < 0$, start with 1000 0000

④a Add num to 128
Calculate difference seen in 3a

$n = -27$

$128 - 27 = 101$

④b Calculate 101 in binary.

$-2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$

101 \rightarrow 0 1 1 0 0 1 0 1

+ -128 \rightarrow 1 0 0 0 0 0 0 0

$-27 \rightarrow$ 1 1 0 0 1 0 1

$$\begin{array}{r} 101 \\ - 64 \leftarrow 2^6 \\ \hline 37 \\ 32 \leftarrow 2^5 \\ \hline 5 \\ 4 \leftarrow 2^2 \\ \hline 1 \leftarrow 2^0 \end{array}$$



Conditional Statements



```
if (condition 1) {  
    // code  
} else if(condition 2) {  
    // code  
} else if(condition 3) {  
    // code  
} else {  
    // code  
}
```




Nested Conditional Statements



```
if (condition 1) {  
    if (inner cond 1) {  
        // code  
    } else {  
        // code  
    }  
} else {  
    // code  
}
```



While Loops



```
int x = 1;
while (x < 10) {
    System.out.println(x);
    x++;
}
```



```
while (outer condition) {  
    while (inner condition) {  
        // code  
    }  
}
```



For Loops



```
for (int i = 1; i <= 2000; i *= 2) {  
    for (int j = 1; j <= i; j *= 2) {  
        System.out.println(i);  
    }  
}
```



Strings



Immutability

Strings cannot be changed nor removed from, but they can be "concatenated" (added) to.



length()

We can get the length of a string using the `.length()` method.

```
String str = "Hello World";  
str.length(); // Gives us 11
```



equals()

We cannot check if two Strings are equal using the == operators.

Instead, we must use the `.equals()` method.

Case sensitive

```
String str = "Hello World";  
String str1 = "Hello World";  
str.equals(str1); // true
```



equals()

We cannot check if two Strings are equal using the == operators.

Instead, we must use the `.equals()` method.

Case sensitive

```
String str = "Hello World";  
String str1 = "hello world";  
str.equals(str1); // false
```



equalsIgnoreCase()

Case insensitive method to check equality

```
String str = "Hello World";  
String str1 = "hello world";  
str.equalsIgnoreCase(str1); // true
```




compareTo()

Case sensitive method to compare strings

```
String str = "Hello World";  
String str1 = "hello world";  
str.compareTo(str1); // -32
```



compareToIgnoreCase()

Case insensitive method to compare strings

```
String str = "Hello World";  
String str1 = "hello world";  
str.compareToIgnoreCase(str1); // 0
```



charAt()

Index a string using `charAt(index)`

```
String str = "Hello World";  
str.charAt(0); // 'H'
```



substring()

Get a portion of a string using
`substring(startIndex, endIndex)`

```
String str = "Hello World";  
str.substring(0, 6); // "Hello"
```



Arrays



Array

Collection of same-typed data



Initialization

```
data type[] variable = new data type[size];
```

```
data type[] variable = { val1, val2, val3, ... valN };
```



Initialization

```
int[] nums = new int[10];
```

```
int[] nums = { 1, 1, 2, 3, 5, 8 };
```




Get

Remember, arrays in Java cannot be added to or removed from. Values are either retrieved using indexes or set using indexes.

```
int[] arr = { 1, 2, 3 };  
System.out.println(arr[0]); // 1
```



Set

Remember, arrays in Java cannot be added to or removed from. Values are either retrieved using indexes or set using indexes.

```
int[] arr = { 1, 2, 3 };  
arr[1] = 10;  
  
System.out.println(arr[1]); // 10
```



length

We can get the number of elements in an array using the `length` property of every array.

```
int[] arr = { 7, 2, 8, 4, 1 };  
  
System.out.println(arr.length); //  
gives 5
```



Iteration

```
int[] arr = new int[10];  
for(int i = 0; i < arr.length; i++) {  
    arr[i] = i * i;  
}  
  
for(int i = 0; i < arr.length; i++) {  
    System.out.print(arr[i] + " ");  
}
```

Gives: 0 1 4 9 16 25 36 49 64 81



Iteration

```
int[] arr = new int[10];  
for(int i = 0; i < arr.length; i++) {  
    arr[i] = i * i;  
}  
  
for(int num : arr) {  
    System.out.print(num + " ");  
}
```

Gives: 0 1 4 9 16 25 36 49 64 81



2D Arrays



2D Arrays

Conceptually, an array inside of an array.



Initialization

```
data type[][] variable = new data type[row size][col size];  
data type[][] variable = { { val1, val2, }, { val3, ... valN } };
```




Initialization

```
int[][] nums = new int[3][5];
```

```
int[][] nums = { { 1, 2, 3 }, { 4, 5, 1 } };
```



Accessing values

Similar to a regular 1D array, we can only get and set values using indexes.

```
int[][] arr = new int[6][6];  
  
System.out.println(arr[1][2]); //  
gives 0  
  
arr[1][2] = 3;  
  
System.out.println(arr[1][2]); //  
gives 3
```



arr.length

Using `length` on the array variable will give you the number of rows.

```
int[][] arr = new int[3][8];  
  
System.out.println(arr.length); //  
Gives 3
```



arr[0].length

However, if you use `length` on an indexed value, it will give you the number of columns.

```
int[][] arr = new int[3][8];  
  
System.out.println(arr[0].length);  
// Gives 8
```



Static Functions



Static Functions

Must have a return type and a name.

```
privacy static return type function name(parameters)
```



Static Functions

```
public static void sum(int[] arr){  
    int total = 0;  
    for(int num : arr) {  
        total += num;  
    }  
    System.out.println(total);  
}
```



Static Functions

```
private static int sum(int[] arr){  
    int total = 0;  
    for(int num : arr) {  
        total += num;  
    }  
    return total;  
}
```




Midterm Review

