CSE1322 J03 Week 1

Variables, Assignment, Data Types, and I/O

Primitive Data Types

- We call the following Data Types primitive because they are simple data that can represent a single basic values.
- These Data Types have fixed sizes which determine their range, these are the following

Data type	Range	Size
byte	-128 to 127	1 byte
short	-32,768 to 32,767	2 bytes
int	-2,147,483,648 to 2,147,483,647	4 bytes
long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	8 bytes
float	~3.4 x 10^38 to 3.4 x 10^38 (~6-7 decimal digits)	4 bytes
double	~1.7 x 10^308 to 1.7 x 10^308 (~15 decimal digits)	8 bytes
char	Unicode characters (0 to 65,535 or '\u0000' to '\uffff')	2 bytes
boolean	`true` or `false` values	1 bit

Complex Data Types

- Complex Data Types are objects. They can hold multiple primitive values, functions, and more complex data types.
- One common example of a Complex Data Types are the `String` class and `Scanner` class.

Variables

- To initialize a variable in Java, you must specify its Data Type and its Identifier. A variable's initial value is optional.
- To initialize a variable with an initial value you must use the assignment symbol `=`. Furthermore, in cases of Complex Data Types you must also use the `new` keyword to construct the object.

```
Dog d1 = new Dog("Rover");
```

I/O operations

- In java, print statements use the `System.out.println()` or `System.out.prin()` functions to print or display text and values into the console.

```
System.out.print("This prints a string, but does not leave a new line after it has been printed. ");
System.out.println("While this print statement will leave a new line after it has been printed");
System.out.print("See?");
```

- To take an input from the console, we need to create a scanner object.
- To make use of the Scanner object, we must import the proper library `import java.util.Scanner;`

```
Scanner sc = new Scanner(System.in); //System.in is used to take in the keyboard input from the console
```

- Furthermore, to use the `Scanner` object to take an input and store it into a variable we must use some of the functions from the `Scanner` class.

  Here are some useful ones:

short	`sc.nextShort()`
int	`sc.nextInt()`
long	`sc.nextLong()`
float	`sc.nextFloat()`
double	`sc.nextDouble()`
boolean	`sc.nextBoolean()`
char	`sc.next.chatAt(0)`
String	`sc.nextLine()`

```
int x;
x = sc.nextInt();
```

Selection Structures

- If/else if/else statements

```
if(x > 10){
  ...
}
else if(x > 5){
  ...
}
else{
  ...
}
```

- Switch case statements

```
switch(x){
  case 1:
    ...
    break;

  case 2:
    ...
    break;

  default:
    ...
}
```

Repetition Structures

- For loop

```
for(int i = 0; i < 10; i++){
  ...
}
```

- While loop

```
while(i < 10){
  ...
  i++;
}
```

- Do-While loop

```
do{
  ...
  i++;
} while(i < 10);
```

Arrays

- Most likely, arrays are the first data structure you have been introduced to.
- Arrays allows you to store data (either primitive or complex) organized under indexes within the same identifier.
- In Java, we start counting the indexes from 0.
- Arrays are a good way to organize data, since it allows you to use repetition structures to traverse it.
- Otherwise, arrays are limited. One of these limitations being that arrays are fixed size. This means that an array cannot change size in runtime.
- Here are some examples of 1D and 2D arrays:

```
int[] array = new int[2];

// Insertion
array[0] = 1;
array[1] = 2;
array[2] = 3;

// Print
for(int i = 0; i < array.length; i++){
  System.out.println(aray[i]);
}
```

```
Dog[][] arrayOfDogs = new Dog[1][1];

// Assuming you fill the 2D array with Dog objects.

// Print
for(int i = 0; i < arrayOfDogs.length; i++){
  for(int j = 0; j < arrayOfDogs[0].length; j++){
    System.out.println(arrayOfDogs[i][j].getName());
  }
}
```

Methods

- Methods are a block of code that performs a specific task or set of tasks.

- Methods helps us encapsulating some of the functionalities of our code, promote reusable code, and it makes our code be more organized.

```
/**
 * Calculates and returns the area of a rectangle.
 *
 * This method takes the length and width of a rectangle as parameters
 * and calculates the area by multiplying the length and width values.
 * The result represents the total area of the rectangle.
 *
 * @param length The length of the rectangle.
 * @param width The width of the rectangle.
 * @return The calculated area of the rectangle.
 */

public static int getArea(int length, int width){
  return length * width;
}
```

```
/**
 * Prints the names of dogs in a 2D array of Dog objects.
 *
 * This method takes a 2D array of Dog objects as input and iterates through
 * the array to print the names of the dogs. It uses nested loops to traverse
 * rows and columns of the array and accesses each Dog object to retrieve
 * its name using the `getName()` method.
 *
 * @param array A 2D array containing Dog objects.
 */

public static void printDogArray(Dog[][] array){
  for(int i = 0; i < arrayOfDogs.length; i++){
    for(int j = 0; j < arrayOfDogs[0].length; j++){
      System.out.println(arrayOfDogs[i][j].getName());
    }
  }
}
```

Classes and Objects

- Classes and Objects are the fundamental concepts for Object Oriented Programming.
- You can think of Classes as a blueprint or template to create objects from.
- While objects are the instances created based on a class.

```
class Car {
  private String brand;
  private String model;

  // Constructor
  Car(String brand, String model) {
    this.brand = brand;
```

```
    this.model = model;
  }

  void startEngine() {
    System.out.println("Engine started!");
  }

  void stopEngine() {
    System.out.println("Engine stopped.");
  }

  // Getter for brand
  public String getBrand() {
    return brand;
  }

  // Setter for brand
  public void setBrand(String carBrand) {
    brand = carBrand;
  }

  // Getter for model
  public String getModel() {
    return model;
  }

  // Setter for model
  public void setModel(String carModel) {
    model = carModel;
  }
}
```

```
public class ExampleA{
  public static void main(String[] args){
    Car myCar = new Car("Toyota", "Camry");
    myCar.startEngine();

    System.out.println(myCar.getBrand());

    myCar.setBrand("Honda");
    System.out.println(myCar.getBrand());
  }
}
```

ExampleA.java