JAVA FULL SYUDY

Day 3:

1. Method

```
public class Main {

    public static void main(String[] args) {

        // method = a block of code that is executed whenever it is called
upon

        int x = 3;
        int y = 4;

        int z = add(x,y);

        System.out.println(z);
    }

    static int add(int x, int y) {

        int z = x + y;
        return z;

    }

}
```

2. Overloaded Method

```
public class Main {

    public static void main(String[] args) {

        // overloaded methods = methods that share the same name but have
different parameters
        //                      method name + parameters = method signature

        double x = add(1.0,2.0,3.0,4.0);

        System.out.println(x);

    }

    static int add(int a, int b) {
        System.out.println("This is overloaded method #1");
        return a + b;
    }
    static int add(int a, int b, int c) {
```

```
        System.out.println("This is overloaded method #2");
        return a + b + c;
    }
    static int add(int a, int b, int c, int d) {
        System.out.println("This is overloaded method #3");
        return a + b + c + d;
    }
    static double add(double a, double b) {
        System.out.println("This is overloaded method #4");
        return a + b;
    }
    static double add(double a, double b, double c) {
        System.out.println("This is overloaded method #5");
        return a + b + c;
    }
    static double add(double a, double b, double c, double d) {
        System.out.println("This is overloaded method #6");
        return a + b + c + d;
    }
}
```

3. Printf

```
public class Main {

    public static void main(String[] args) {

        // printf() =   an optional method to control, format, and display
text to the console window
        //              two arguments = format string +
(object/variable/value)
        //              % [flags] [precision] [width] [conversion-character]

        boolean myBoolean = true;
        char myChar = '@';
        String myString = "Bro";
        int myInt = 50;
        double myDouble = 1000;

        // [conversion-character]
        //System.out.printf("%b",myBoolean);
        //System.out.printf("%c",myChar);
        //System.out.printf("%s",myString);
        //System.out.printf("%d",myInt);
        //System.out.printf("%f",myDouble);

        //[width]
        // minimum number of characters to be written as output
        //System.out.printf("Hello %10s",myString);

        //[precision]
        // sets number of digits of precision when outputting floating-point
values
```

```
        //System.out.printf("You have this much money %.1f",myDouble);

        // [flags]
        // adds an effect to output based on the flag added to format
specifier
        // - : left-justify
        // + : output a plus ( + ) or minus ( - ) sign for a numeric value
        // 0 : numeric values are zero-padded
        // , : comma grouping separator if numbers > 1000

        //System.out.printf("You have this much money %,f",myDouble);
    }
}
```

4. final keyword

```
public class Main {

    public static void main(String[] args) {

        // 被final定义后的variable不能更改
        final double PI = 3.14159;

        PI = 4;

        System.out.println(PI); // display: 3.14159

    }
}
```

4. Java objects(OOP)

- object = an instance of a class that may contain attributes and methods
- example: (phone, desk, computer, coffee cup)

```
//*****************************************
public class Main {

    public static void main(String[] args) {

        // We can create multiple instance for the same class
        Car myCar1 = new Car(); // it has all attributes in car cass, and
the drive and brake methods.
        Car myCar2 = new Car();

        System.out.println(myCar1.make); //access the attributes
        System.out.println(myCar1.model);

        // To perform the function
        // name.method(); then we can use the method
        myCar1.drive();
        myCar1.brake();
    }
```

```
}
//*****************************************
public class Car {

    String make = "Chevrolet";
    String model = "Corvette";
    int year = 2020;
    String color = "blue";
    double price = 50000.00;

    void drive() {
        System.out.println("You drive the car");
    }
    void brake() {
        System.out.println("You step on the brakes");
    }
}
//*****************************************
```

5. Constructors
    - constructor = special method that is called when an object is instantiated(created)

```
public class Main {

    public static void main(String[] args) {

        // Define an object for Human class, named human1
        // human 1 can set its attributes by constructor
        // human 1 can access the method in class by human1.method
        Human human1 = new Human("Rick",65,70);
        Human human2 = new Human("Morty",16,50);

        human1.drink();
        human2.eat();

    }
}
//*******************************************************
public class Human {

    String name;
    int age;
    double weight;

    // Constructor,same name as the class
    // set the attributes for the object
    // it is convenient to set attributes for different object
    Human(String name,int age,double weight){

        // this actually points to the specific instance
        // for example, for the human 1 call
        // this points to human 1
        // this.name->Rick, this age->65, this.weight->70
        this.name = name;
        this.age = age;
```

```
        this.weight = weight;
    }

    void eat() {
        // print the person's name+"is eating"
        // display: rick is eating
        System.out.println(this.name+" is eating");
    }
    void drink() {
        System.out.println(this.name+" is drinking *burp*");
    }

}
//*****************
```

6. local and global variables
    - local = declared inside a method
      // visible only to that method
    - global = declared outside a method, but within a class
      // visible to all parts of a class

```
//*********************************************
public class Main {

 public static void main(String[] args) {

  // Define an object for diceRoller class, named diceRoller
  DiceRoller diceRoller = new DiceRoller();

 }
}
//*********************************************
import java.util.Random; // import Random class

public class DiceRoller {
 // Declare random and number outside the constructor
 // roll ()  can acess these two variables now
 Random random;
 int number;

 DiceRoller(){
  random = new Random();
  roll();
 }

 void roll() {
  // limit the random number size within 6
  // As computer always generate index
  // we need to add one to present the number on the dice
  number = random.nextInt(6)+1;
  System.out.println(number);
 }
}
//*********************************************
```

```
// Or we can pass the local variable to another method
public class DiceRoller {


 DiceRoller(){
  random = new Random();
  int number=0;
  roll(random,number)
 }

 void roll(Random random, int number) {
  // limit the random number size within 6
  // As computer always generate index
  // we need to add one to present the number on the dice
  number = random.nextInt(6)+1;
  System.out.println(number);
 }
}
```

7. Overloaded Constructor

```
public class Main {

    public static void main(String[] args) {


        Pizza pizza = new Pizza("thicc
crust","tomato","mozzerella","pepperoni");

        System.out.println("Here are the ingredients of your pizza: ");
        System.out.println(pizza.bread);
        System.out.println(pizza.sauce);
        System.out.println(pizza.cheese);
        System.out.println(pizza.topping);

    }
}
//*************************************************************************
******
public class Pizza {


    String bread;
    String sauce;
    String cheese;
    String topping;

    Pizza(){

    }

    Pizza(String bread){

        this.bread = bread;
```

```
    }

    Pizza(String bread,String sauce){

        this.bread = bread;
        this.sauce = sauce;
    }

    Pizza(String bread,String sauce,String cheese){

        this.bread = bread;
        this.sauce = sauce;
        this.cheese = cheese;
    }

    Pizza(String bread,String sauce,String cheese,String topping){

        this.bread = bread;
        this.sauce = sauce;
        this.cheese = cheese;
        this.topping = topping;
    }
}//*************************************************************************
******
```

8. to-String Method

- toString() = special method that all objects inherit, that returns a string that "textually represents" an object
- can be used both implicitly and explicitly

```
public class Main {

 public static void main(String[] args) {



  Car car = new Car();

  // 当我们想print出 car object的性质
  // 我们需要很多行println去指向它
  // 有to string method之后我们只需要一条println，将所有attribute放在tostring里面
  System.out.println(car.toString());// Display:  Ford
//                                                Mustang
//                                                 red
 //                                                2021

  // or

  System.out.println(car); // 单独用是不对的，将display the address of car
object。然而define tostring method 之后可以直接print car，是一种tostring的隐形用
法。


 }
}
```

```
//****************************************
public class Car {


 String make = "Ford";
 String model = "Mustang";
 String color = "red";
 int year = 2021;

 // Method 1:
 public String toString() {

  return make +"\n"+model+"\n"+color+"\n"+year;

 }

 // Method 2:
  public String toString() {

     String mystring= make+"\n"+"\model"+"\n"+color;
     return mystring;

 }
}
//****************************************
```

9. Array of Object

```
public class Main {

    public static void main(String[] args) {
        //普通定义array的方式: datatype[] name = new datatype[arraySize];

        //Food[] refrigerator = new Food[3];

        Food food1 = new Food("pizza");
        Food food2 = new Food("hamburger");
        Food food3 = new Food("hotdog");

        //datatype[] name = new datatype[arraySize]
        Food[] refrigerator = {food1,food2,food3};

        //refrigerator[0] = food1;
        //refrigerator[1] = food2;
        //refrigerator[2] = food3;

        System.out.println(refrigerator[0].name);
        System.out.println(refrigerator[1].name);
        System.out.println(refrigerator[2].name);

    }
}
//*******************************************************
public class Food {
```

```
    String name;

    Food(String name){
        this.name = name;
    }
}//*****************************************************
```

10. Object Passing
    - Pass object as an argument

```
//*****************************************
public class Main {

    public static void main(String[] args) {

        Garage garage = new Garage();

        Car car1 = new Car("BMW");
        Car car2 = new Car("Tesla");

        garage.park(car1);
        garage.park(car2);

    }
}
//*****************************************
public class Garage {

    // The parameter has to set to the datatype
    void park(Car car) {
        System.out.println("The "+car.name+" is parked in the garage");
    }
}//*****************************************
public class Car {


    String name;

    Car(String name){
        this.name = name;
    }

}}//
```