Declaration: String name;
Initialization: name = “Cindy”;
Const variables: variable to be all caps. final double TAX = 0.25;
Polymorphism: many forms of the same method
Passing the instance object in as a parameter

DML

Instance variables Private int empID = -empID: int

Methods

- Returns something | Public int add(int a, int b) +add(a:int, b:int):int
- Void method Public setNum(int a) +setNum(a:int)

Constructor

```
Public Person(String name){                    +Person(name:String)
        Name = name;
}
```

- Class(template) | Object(instance of the class)
- static(belongs to the class) | non-static(belongs to the instance of the class)
- Constructor(template for instantiating the class)
- @Overriding methods(subclass can override any super class’s methods)
- Overload method: method with the same method name, but different parameter lists

Scanner Class:

```
        Import java.util.Scanner;                          // package
        Scanner userInput = new Scanner(System.in);        // Scanner instance
        Int id = userInput.nextInt();           // prevent computer from skipping keyboard inputs
        userInput.nextLine();                          // prevent computer from skipping keyboard inputs
```

Decimal formatting:

- 1. Use the printf method:
    - double price = 34.12;
    - System.out.printf(“%.2f”, price); // prints 34.12
- 2. Use DecimalFormat:
    - Import java.text.DecimalFormat;
    - double price = 34.12;
    - DecimalFormat dc = new DecimalFormat(“#00.00”);
    - System.out.println(dc.format(price));

For - Loop structure:

```
        Example: for(int counter=0; counter<100; counter++){
                        System.out.println(“Hello World”);}
```

While - Loop structure: (checks condition first, can run zero times)

```
        Example: Int num = 0;
                while( num <=10){
                        System.out.println(“Hello World”);
                        Num++;}
```

Switch Statement:

```
        Example: Int num = userInput.nextChar();
                 userInput.nextLine();
                 switch(num){
                        Case 1:  System.out.println(“You’ve entered the number 1”);
                                 break;
                        Case 2: System.out.println(“You’ve entered the number 2”);
                                 break;
                        Default: System.out.println(“You didn’t enter a number?”);  }
```

If-Statements: if(condition){

```
                // statements
              } else if(condition){
                // statements }
```

Logical Operators:

- OR || : evaluates to true if one of the condition is true
- AND && : evaluate to true if all condition is true
- NOT ! : converts to opposite condition

Inheritances: (extend keyword)

- Form of software use in which a new class is created by absorbing an existing class's members. The new class can add/modify capabilities to the original class
- Variables and methods of the parent class are included in the child class by inheritance. Additional members are added to the child in its class definition
- Inheritance is called a **is-a** relationship between class
- superclass(existing class/top) | subclass(new class inheriting the superclass)
- Parent -> child | base -> derived | superclass -> subclass
- Direct superclass: superclass from which the subclass explicitly inherits
- Indirect superclass: any class above the direct superclass in the class hierarchy
- Object: the class at the top of the java class hierarchy
- Protected: more access than private, but less than public
- **Is-a**: represents inheritance, an object of a subclass can be treated as a object of the super class
- **Has-a**: represents composition, an object contains as members references to other objects
- Example: // subclass's new instance variables are added to the constructor like this:

```
Public class Person(){
        Private String firstName;
        Public Person(String name){
        this.firstName = name;  }}
Public class Employee extends Person{
        Private String lastName;
        Public Employee(String name, String lastName){
                        super(name);
                        this.lastName = lastName;   }}
Public class Faculty extends Employee{
        Private String schoolName;
        Public Faculty(String firstName, String lastName, String schoolName){
                super(firstName, lastName);
                this.schoolName = schoolName;   }}
```

instanceof operator: test whether an object is an instance of a class

```
Example: if(a instanceof Bird){
        System.out.println(" a is a instance of bird!")'}
```

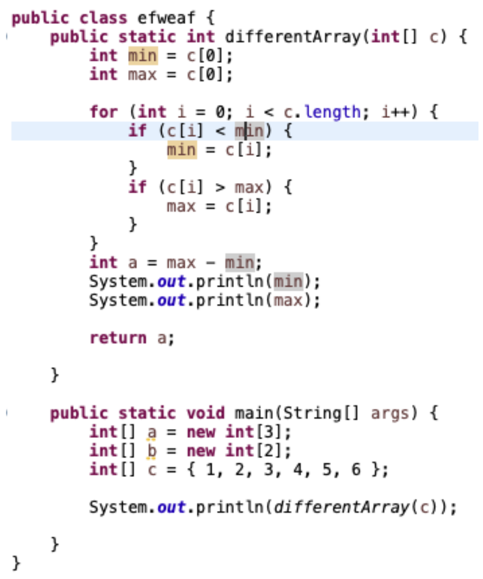
Quiz questions:

The variables defined in the method header are known as actual parameters.   False

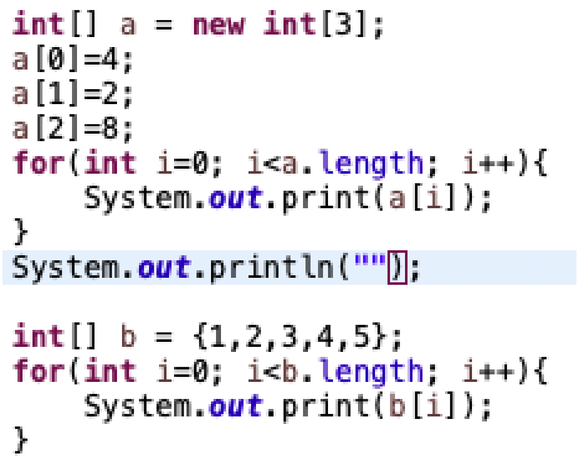
When a method is invoked, you pass a value to the parameter. This value is referred to a: actual parameter

Constructors are inherited in a inheritance relationship: false

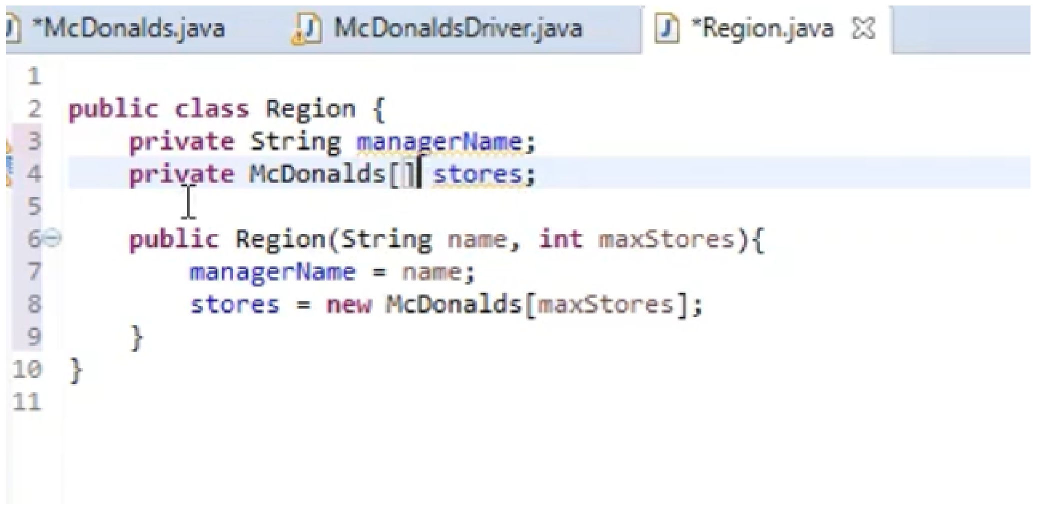
Finding the difference of array: declaring array(two types):



```java
public class efweaf {
    public static int differentArray(int[] c) {
        int min = c[0];
        int max = c[0];

        for (int i = 0; i < c.length; i++) {
            if (c[i] < min) {
                min = c[i];
            }
            if (c[i] > max) {
                max = c[i];
            }
        }
        int a = max - min;
        System.out.println(min);
        System.out.println(max);

        return a;

    }

    public static void main(String[] args) {
        int[] a = new int[3];
        int[] b = new int[2];
        int[] c = { 1, 2, 3, 4, 5, 6 };

        System.out.println(differentArray(c));

    }
}
```



```java
int[] a = new int[3];
a[0]=4;
a[1]=2;
a[2]=8;
for(int i=0; i<a.length; i++){
    System.out.print(a[i]);
}
System.out.println("");

int[] b = {1,2,3,4,5};
for(int i=0; i<b.length; i++){
    System.out.print(b[i]);
}
```



```java

public class Region {
    private String managerName;
    private McDonalds[] stores;

    public Region(String name, int maxStores){
        managerName = name;
        stores = new McDonalds[maxStores];
    }
}

```

Array equals: System.out.println(Arrays.equals(a,b));