Declaration: String name; Initialization: name = “Cindy”;

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;

- static(belongs to the class) | non-static(belongs to the instance of 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
userInput.next();                      // uses whitespace as default delimiter, returns tokenized text
userInput.nextLine();                  // returns all text up to a line break
```

Decimal formatting:

- 1. Use the printf method: System.out.printf(“%.2f”, 34.12); // prints 34.12

Switch Statement:

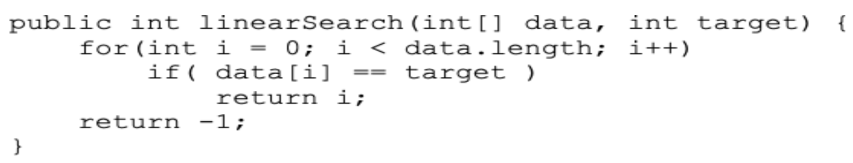
```
Example: Int num = userInput.nextInt();
         userInput.nextLine();
         switch(num){
                Case 1:  System.out.println(“You’ve entered the number 1”);
                         break;
                Default: System.out.println(“You didn’t enter a number?”); }
```

Inheritances: (extend keyword)

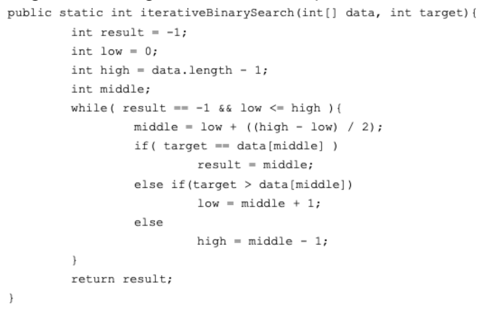
- Form of software use in which a new class is created by absorbing an existing class’s members. The new class can add/modify capabilities to the original class
- **Is-a**: represents inheritance, an object of a subclass can be treated as a object of the super class
- **Has-a**: represents composition, 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;   }}
```

Linear Search: used when list isn’t sorted, algorithm: start at the first item, is it the one I’m looking for? If not go to next item, repeat until found or items are checked



```
public int linearSearch(int[] data, int target) {
    for(int i = 0; i < data.length; i++)
        if( data[i] == target )
            return i;
    return -1;
}
```



```
public static int iterativeBinarySearch(int[] data, int target){
        int result = -1;
        int low = 0;
        int high = data.length - 1;
        int middle;
        while( result == -1 && low <= high ){
                middle = low + ((high - low) / 2);
                if( target == data[middle] )
                        result = middle;
                else if(target > data[middle])
                        low = middle + 1;
                else
                        high = middle - 1;
        }
        return result;
}
```

Binary Search: used on sorted arrays, algorithm: start at the middle, is the middle equal to target? If its less, then move to the right side of array/ If its greater, move to the left side of the array

isSorted method: takes an array as input and returns whether the array is sorted based on true/false

```
public static boolean isSorted(int[] arr){ boolean output = true;
            for (int i=0 ; i< arr.length - 1; i++)
            if (arr[i] > arr[i+1])output = false;
            return output;}
```

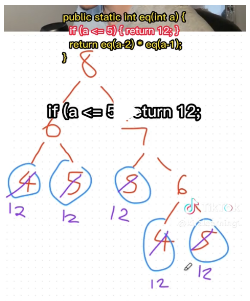
Selection Sort Code:
```
public static void selectionSort( int[] array ){
                  for ( int j=0; j<array.length-1; j++ ){
                  int min = j;
                  for ( int k=j+1; k<array.length; k++ )
                  if ( array[k] < array[min] )
                  min = k;
                  int temp = array[j];
                  array[j] = array[min];
                  array[min] = temp;   }}
```

Selection Sort with Strings(compareTo):
```
Public static void selectionSort( String[] array){
                  for(int j=0;j<array.length-1;j++){
                         intmin=j;
                  for(int k=j+1; k<array.length;k++)
                  if(array[k].compareTo(array[min])<0)min=k;
                  String temp = array[j];
                  Array[j] = array[min];
                  Array[min]  = temp;}}
```

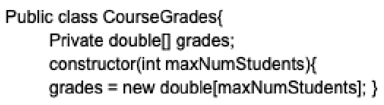
PrintWriter: used to send characters to a text file ex: PrintWriter output = new PrintWriter(“myOutput.txt”); output.println(“Hello World”);

Exception:
```
try{
            // method 1
            } catch(ExceptionType ex){
                    System.out.println(“exception here”);
            } finally {
                    System.out.println(“end of line”); }
```

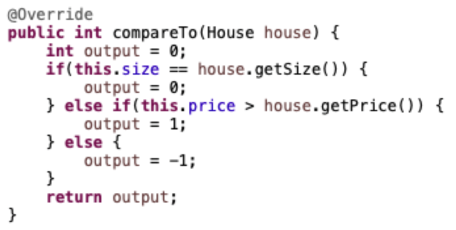
Recursion: technique that solves a problem by solving a smaller problem of the same type
- Base case: a problem that can be solved immediately
- Decomposition: smaller identical problems
- Composition: smaller problems answers combined to form the answers of large problem



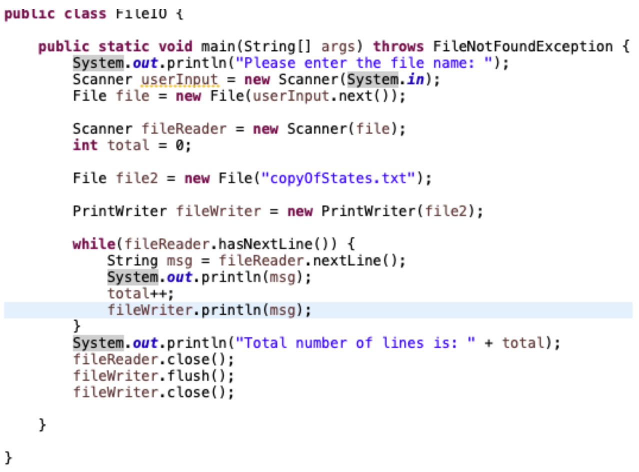
```
public static int eq(int a) {
if (a <= 5) { return 12; }
return eq(a-2) * eq(a-1);
}
```



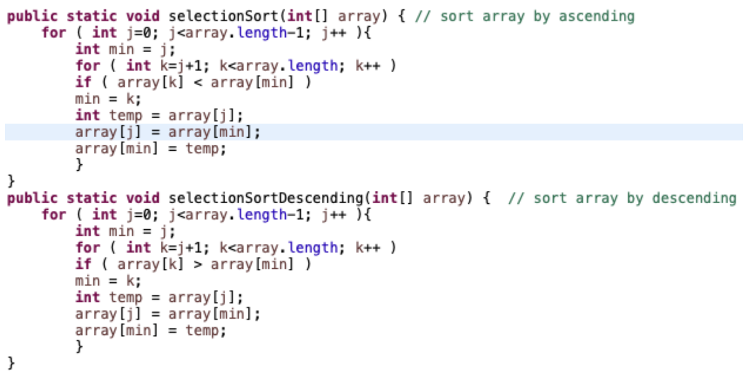
Public class CourseGrades{
Private double[] grades;
constructor(int maxNumStudents){
grades = new double[maxNumStudents]; }



```
@Override
public int compareTo(House house) {
    int output = 0;
    if(this.size == house.getSize()) {
        output = 0;
    } else if(this.price > house.getPrice()) {
        output = 1;
    } else {
        output = -1;
    }
    return output;
}
```



```
public class FileIO {

    public static void main(String[] args) throws FileNotFoundException {
        System.out.println("Please enter the file name: ");
        Scanner userInput = new Scanner(System.in);
        File file = new File(userInput.next());

        Scanner fileReader = new Scanner(file);
        int total = 0;

        File file2 = new File("copyOfStates.txt");

        PrintWriter fileWriter = new PrintWriter(file2);

        while(fileReader.hasNextLine()) {
            String msg = fileReader.nextLine();
            System.out.println(msg);
            total++;
            fileWriter.println(msg);
        }
        System.out.println("Total number of lines is: " + total);
        fileReader.close();
        fileWriter.flush();
        fileWriter.close();

    }

}
```



```
public static void selectionSort(int[] array) { // sort array by ascending
    for ( int j=0; j<array.length-1; j++ ){
        int min = j;
        for ( int k=j+1; k<array.length; k++ )
        if ( array[k] < array[min] )
        min = k;
        int temp = array[j];
        array[j] = array[min];
        array[min] = temp;
        }
}
public static void selectionSortDescending(int[] array) {  // sort array by descending
    for ( int j=0; j<array.length-1; j++ ){
        int min = j;
        for ( int k=j+1; k<array.length; k++ )
        if ( array[k] > array[min] )
        min = k;
        int temp = array[j];
        array[j] = array[min];
        array[min] = temp;
        }
}
```