## Exercise 13.3

1. **static** **int** raise(**int** base, **int** expo){

**if** (expo== 0)

**return** 1;

**else** **if** (expo % 2== 0)

**return** base \* *raise*(base,expo/2);

**else**

**return** base \* *raise*(base, expo -1);

1. 2 \* *raise*(2,3)=8

2\* raise (2, 2) =4 == 16

2 \* raise (2, 0) =2

1

## Exercise 13.4

1. 1) Choose the pivot randomly 2) Use the median of the array 3) Use the first element as pivot
2. Can’t only divide a array in half so many times , will arrive at a single element…. Worst case: O(n2 )

## Review questions pg. 532

1. Stopping state
2. Recursive step
3. Call stack
4. Activation record
5. Infinite recursion
6. Stack overflow error

## Labs

//© A+ Computer Science - www.apluscompsci.com

//Name -Josh Martin

//Date -03/30/2015

//Class -

//Lab - Lab 1

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**import** java.util.Collections;

**import** **static** java.lang.System.\*;

**public** **class** Lab1

{

**public** **static** **void** main( String args[] ) **throws** IOException

{

Scanner file = **new** Scanner(**new** File("lab1.dat"));

ArrayList<Word> words = **new** ArrayList<Word>();

**while**(file.hasNext())

{

//add in a new Word to words

words.add(**new** Word(file.next()));

}

**for** (**int** i = 0; i < words.size()-1 ; i++)

{

**int** min = i;

**for**(**int** j= i+1; j < words.size(); j++){

**if**((words.get(min)).compareTo(words.get(j)) > 0)

min =j;

}// end of inner loop

**if** (min!= i){

Word temp = words.get(min);

words.set(min,words.get(i));

words.set(i,temp);

}// end of sort if

} //end of outer loop

//print out words

**for** (**int** j =0; j < words.size(); j++)

System.out.println(words.get(j));

}

}// end of class

//© A+ Computer Science - www.apluscompsci.com

//Name - Josh Martin

//Date - 03/30

//Class -

//Lab - Lab 1

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**import** java.util.Collections;

**import** **static** java.lang.System.\*;

**public** **class** Word **implements** Comparable<Word>

{

**private** String word;

//constructors

**public** Word()

{

word = "";

}

**public** Word(String a)

{

word =a;

}

//compareTo

**public** **int** compareTo(Word other)

{

**int** dif = word.length() - other.word.length();

**if** (dif == 0)

**return** word.compareToIgnoreCase(other.word);

**return** dif;

}

//toString

**public** String toString()

{

**return** word;

}

}// end of class

//© A+ Computer Science - www.apluscompsci.com

//Name -Josh Martin

//Date -

//Class -

//Lab - Lab 2

**import** java.util.Arrays;

**import** java.util.Scanner;

**import** **static** java.lang.System.\*;

**import** **static** java.util.Arrays.\*;

**import** java.util.ArrayList;

//define the Grade class

**public** **class** Grade{

//add in a instance variable of type double named grade

**double** grade;

//constructors - default and initialization

**public** Grade()

{

grade = 0.0;

}

**public** Grade(**double** a)

{

grade = a;

}

//method setGrade sets the grade

**public** **void** setGrade(**double** a)

{

grade = a;

}

//method getNum returns the grade

**public** **double** getNum()

{

**return** grade;

}

//method getLetterGrade returns a String A, B, C, D, or F

**public** String getLetterGrade()

{

**if** (grade >=89.5)

**return** "A";

**if** (grade >= 79.5)

**return** "B";

**if** (grade >= 69.5)

**return** "C";

**if** (grade >= 59.5)

**return** "D";

**return** "F";

}

//method toString returns grade and the letter grade

**public** String toString()

{

**return** grade + " - " + getLetterGrade();

}

}//end of class

//© A+ Computer Science - www.apluscompsci.com

//Name -Josh Martin

//Date -

//Class -

//Lab - Lab 2

**import** java.util.Arrays;

**import** java.util.Scanner;

**import** **static** java.lang.System.\*;

**import** **static** java.util.Arrays.\*;

**import** java.util.ArrayList;

**public** **class** Grades

{

**private** ArrayList<Grade> grades;

**public** Grades()

{

grades= **new** ArrayList<Grade>();

}

**public** Grades(String gradeList)

{

// finds - and adds one soo it can find the second space

**int** oldSpace = gradeList.indexOf("-") + 1;

String[] temp = gradeList.substring(oldSpace+ 1).split(" ");

// gets how many items are in the array list

**int** len = gradeList.charAt(0);

grades= **new** ArrayList<Grade>(len);

**for** (**int** i =0; i < temp.length; i++)

{

**double** temp2 =Double.*parseDouble*(temp[i]);

grades.add(**new** Grade(temp2));

}

}

// do I put the same thing as above?

**public** **void** setGrades(String gradeList)

{ String[] temp =gradeList.split(" ");

**for** (**int** i = 0; i < grades.size();i++)

grades.set(i,**new** Grade(Double.*parseDouble*(temp[i])));

}

**public** **void** setGrade(**int** spot, **double** grade)

{

grades.get(spot).setGrade(grade);

}

**public** **double** getSum()

{ **double** sum =0;

**for** (**int** i= 0; i<grades.size();i++)

sum+=grades.get(i).getNum();

**return** sum;

}

**public** **double** getLowGrade()

{

**double** lowest=0;

//selection sort to find the lowest number in the list

**for** ( **int** i = 0; i< grades.size()-1;i++)

{

**int** min =i;

**for**(**int** j=i+1; j < grades.size(); j++)

{

**if**(grades.get(j).getNum() < grades.get(min).getNum())

min=j;

}// end of outer

**if** (min!= i)

lowest = grades.get(min).getNum();

}//end of outer loop

**return** lowest;

}

**public** **double** getHighGrade()

{

**double** lowest=0;

//selection sort to find the Highest number in the list

**for** ( **int** i = 0; i< grades.size()-1;i++)

{

**int** min =i;

**for**(**int** j=i+1; j < grades.size(); j++)

{

**if**(grades.get(j).getNum() > grades.get(min).getNum())

min=j;

}// end of outer

**if** (min!= i)

lowest = grades.get(min).getNum();

}//end of outer loop

**return** lowest;

}

**public** **void** removeValue()

{ **int** find =0;

**for**(**int** i =0; i< grades.size();i++)

**if** (grades.get(i).getNum()== getLowGrade())

find= i;

grades.remove(find);

}

**public** **int** getNumGrades()

{

**return** grades.size();

}

**public** String toString()

{

String output="";

**for** (**int** i =0; i<grades.size(); i++)

output += grades.get(i).toString() + " ";

**return** output;

}

}// end of class

//© A+ Computer Science - www.apluscompsci.com

//Name - Josh martin

//Date -

//Class -

//Lab - Lab 2

**import** java.util.Scanner;

**import** **static** java.lang.System.\*;

**import** java.lang.Comparable;

**public** **class** Student **implements** Comparable<Student>

{

**private** String myName;

**private** Grades myGrades;

//add in constructors

**public** Student()

{

myName ="";

}

**public** Student(String a, String b){

myName =a;

myGrades = **new** Grades(b);

}

//add in set methods

//add in get methods as shown in StudentTester

**public** String getName()

{

**return** myName;

}

**public** Grades getGrades()

{

**return** myGrades;

}

**public** **double** getSum(){

**return** myGrades.getSum();

}

**public** **double** getAverage(){

**return** getSum()/myGrades.getNumGrades();

}

**public** **double** getAverageAfterDroppingLow(){

myGrades.removeValue();

**return** getSum()/myGrades.getNumGrades();

}

**public** **double** getLowGrade(){

**return** myGrades.getLowGrade();

}

**public** **double** getHighGrade(){

**return** myGrades.getHighGrade();

}

**public** **int** compareTo(Student other)

{

//method compareTo will compare this Student to the other student

**if** (getAverage()> other.getAverage())

**return** 10;

//method equals will compare this Student to the other student

**if** (getAverage()== other.getAverage())

**return** 0;

**return** -10;

}

//add in the toString

**public** String toString()

{

**return** myName + " = " + myGrades;

}

}// end of class

//© A+ Computer Science - www.apluscompsci.com

//Name - Josh Martin

//Date -

//Class -

//Lab - Lab 2

**import** java.util.Scanner;

**import** **static** java.lang.System.\*;

**import** java.util.Arrays;

**import** java.util.ArrayList;

**import** java.util.Collections;

**public** **class** Class

{

**private** String name;

**private** ArrayList<Student> studentList;

**public** Class()

{

name ="";

studentList = **new** ArrayList<Student>();

}

**public** Class(String names, **int** stuCount)

{

name =names;

studentList = **new** ArrayList<Student>(stuCount);

}

**public** **void** addStudent(**int** stuNum, Student s)

{

studentList.add(stuNum,s);

}

**public** **void** sort()

{

//Bubble sort

**for** ( **int** i = 0; i< studentList.size()-1; i++)

**for** (**int** j = 0; j< studentList.size()-1;j++)

**if**(studentList.get(j).compareTo(studentList.get(j+1))> 0)

{

Comparable temp = studentList.get(j);

studentList.set(j,studentList.get(j+1));

studentList.set(j+1,(Student) temp);

}

}

**public** String getClassName()

{

**return** name;

}

**public** **double** getClassAverage()

{ **double** avg=0;

**int** count=0;

**for** (**int** i =0; i < studentList.size();i++){

avg+=studentList.get(i).getAverage();

count++;

}

**return** avg/count;

}

**public** **double** getStudentAverage(**int** stuNum)

{

**return** studentList.get(stuNum).getAverage();

}

**public** **double** getStudentAverage(String stuName)

{

**for** (**int** i =0; i< studentList.size();i++)

**if**(stuName ==studentList.get(i).getName())

**return** studentList.get(i).getAverage();

**return** -1;

}

**public** String getStudentName(**int** stuNum)

{

**return** studentList.get(stuNum).getName();

}

**public** String getStudentWithHighestAverage()

{

String temp="";

**for**(**int** i=0; i<studentList.size()-1;i++)

{

**int** min=i;

**for**(**int** j=0; j<studentList.size();j++)

{

**if**(studentList.get(j).compareTo(studentList.get(min))> 0)

min =j;

}

**if**(min!=i)

temp =studentList.get(min).getName();

}

**return** temp;

}

**public** String getStudentWithLowestAverage()

{ String temp="";

**for**(**int** i=0; i<studentList.size()-1;i++)

{

**int** min=i;

**for**(**int** j=0; j<studentList.size();j++)

{

**if**(studentList.get(j).compareTo(studentList.get(min))< 0)

min =j;

}

**if**(min!=i)

temp =studentList.get(min).getName();

}

**return** temp;

}

**public** String getFailureList(**double** failingGrade)

{ String str="";

**for** (**int** i= 0; i < studentList.size();i++)

**if** (studentList.get(i).getAverage()<= failingGrade)

str += studentList.get(i).getName() + " ";

**return** str;

}

**public** String toString()

{ String str="";

**for** (**int** i =0; i < studentList.size();i++)

str += studentList.get(i).getName() + " = "+ studentList.get(i).getGrades() +"\n";

**return** name +"\n"+str;

}

}// end of class