**AP CS Midterm Review**

**Unit 1: Computer Systems – Numeric String Rep Number Systems**

Numeric data is stored in two different ways, integer and floating point.

Floating point types suffer from inaccuracies due to the difficulty of storing base 10 fractions using base 2.

Therefore sometimes there are floating point round off errors.

int grade1 = 76, grade2 = 91, grade3 = 99;  
?: How do I store a floating point average of these 3 numbers in double avg?  
double avg = (grade1+grade2+grade3)/3.0;

or ((double)grade1+grade2+grade3)/3;

Strings are arrays of characters. They are an object oriented class in java.

**Unit 2:Math Operators – System Design**

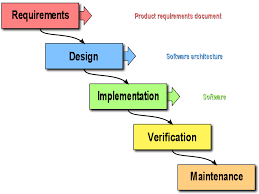
Know the math operators +,-,/ int, / double, %, \*

System Design:

Top Down Programming: start with big picture, then fill out details as you go

Bottom Up Programming: start with idea, build up around it

Extreme Programming: Fast iterations, meet with customer, pair programming, changes often according to customer needs

Waterfall Model: 

Object Oriented Design: Each class has its own area of responsibility

System Testing, Unit Testing, Integration Testing, User Acceptance Testing

**Unit 3:Object Oriented Programming**

Class Design: Designing a class involves identifying variables that represent the state of the object. Methods allow the object to take actions. Constructors initialize an object’s state.

Encapsulation: An object should hide its details and provide other programmers with an interface of methods to manipulate the object.

Interfaces: A set of methods that must be implemented by the class that implements the interface. Interfaces allow a programmer to have standard well known methods that can be called.

You can pass variables of an interface to a method or create variables of an interface in a program.

Comparable Interface: One of the most important interfaces in java is Comparable. It has only one method header, public int compareTo(Object other)

It allows two objects of the same class to be compared mathematically.

Works like the mathematical operator subtraction.

public class Pizza implements Comparable{  
 private String description;  
 private int rating; //1-10, 10 is best

public int getRating() { return rating; }  
 public int compareTo(Object other) { //finish  
 Pizza p = (Pizza)other;

return rating-p.getRating();   
}

Can a class implement multiple interfaces? yes

How about a JFrame that is named MatchingFrame that uses the keyboard and mouse?

public class MatchingFrame extends JFrame implements MouseListener, KeyListener {

// Needs all 5 methods of MouseListener and   
 // all 3 methods of KeyListener

}

Which of the following is allowable?

Pizza p = new Pizza(); OK NOT OK

p.rating = 7; OK NOT OK PRIVATE

Comparable c = new Pizza(); OK NOT OK

c.getRating(); OK NOT OK – c can only use compareTo

Pizza p1 = new Pizza(); OK NOT OK

if(p.compareTo(p1) < 0) { OK NOT OK

System.out.println(“p is lower than p1”);  
}

**Unit 4:Strings**

Strings have many useful methods that allow us to search, break down and compare Strings

indexOf(String target) //-1 if can’t find the target!!!

indexOf(String target, int start)

substring(int c1, int c2) //first char you want, first char you don’t want

substring(int c1) //starts at c1 and takes the rest of the String

String word = “APCS 6 6 7\*”;

Find the index of the second 6

int where = word.indexof(“6”,6); //5 is the first 6

Pull out a substring starting at the first 6 and getting the rest

String newString = word.substring(5);

Search for a \*, print out the index where it occurs

int where = word.indexof(“\*”);

system.out.println(where);

Make a 2nd String that is going to be similar to the original

Loop through every character of the String, if there is a space, put in an \_ in the new String, otherwise copy the original character over

Example: APCS\_6\_6\_7\*

String temp= ””;

for(int i =0; i<word.length();i++){

if(word.substring(i,i+1).equals(“ “){

temp += ”\_”;

else temp += word.substring(i,i+1);

**Unit 5:Arrays ArrayLists**Arrays: Allow storage of multiple variables using one name and a subscript (index)

Size of arrays are fixed at the time of creation

Looping from start of array  
Looping backwards from end of array

Write a method that displays the strings by looping from start of array

public void displayForward(String[]names){  
 for(int i = 0; i<names.length; i++){

System.out.println(names[i]);  
}

Write a method that displays the strings by looping from end of array backwards

public void displayBackward(String[]names){  
 for(int i = names.length-1; i >= 0; i--){  
 System.out.println(names[i]);  
}

Arrays of references

It is possible to use an array to store references to objects

This allows a programmer to store more complex types and control them within the array

Color[]cols = new Color[10]; //creates 10 Color references

Write a loop to initialize the array with 10 colors, fix the red and blue at 128, let the green increase with the array

for(int i = 0; i < cols.length; i++){  
 cols[i] = new Color(128, i\*10, 128);  
}

2d arrays

You can also create a chart or table using a 2d array

String[][] movieNames = new String[10][8];

ArrayList: An ArrayList implements the List interface

There are other classes that implement List, notably the LinkedList

A List has the ability to add, remove, set, get size() etc.

Remember that set changes an element in the list by REPLACING what is at that index

Use .size() to figure out the number of elements in the list

For each loop: A super convenient way to DISPLAY or ACCESS a list or array

DO NOT USE when you are changing or removing from a list or array

for(String s: names)

System.out.println(s); //cycles through from first to last the names in the list

Tracing through code

Make sure you know how to trace through a program, keeping track of the variables;

Example1:

List list1 = new ArrayList<Integer>(); //Integer allows ints to be stored inside an ArrayList

|  |  |  |
| --- | --- | --- |
| list1 | i | OUTPUT |
| 5,6,7,8,9 | 5 | 5  7  9  9 |
|  |  |
| 5,7,8,9 |  |
|  |  |
| 5,7,9,9 |  |
|  |  | 5  100  7  9  9  12 |
| 5,7,9,9,12 |  |
|  |  |
| 5,100,7,9,9,12 |  |
|  |  |
| 1,5,100,7,9,9,12 |  |
|  |  |
| 1 |  |
|  |  |
|  |  |
|  |  | 1 |
|  |  |  |
|  |  |  |
|  |  |  |

for (int i =5; i < 10; i++)

list1.add(i);

list1.remove(1);

list1.set(2,9);

for(Integer x:list1)

System.out.println(x);

list1.add(12);

list1.add(1,100);

for(Integer x:list1)

System.out.println(x);

list1.add(0,1);

//Remove from end

for(int i = list1.size()-1; i>0; i--)

list1.remove(i);

for(Integer x:list1)

System.out.println(x);

|  |  |  |  |
| --- | --- | --- | --- |
| **Bin** | **Oct** | **Dec** | **Hex** |
| 1101 | 15 | 13 | D |
| 10110 | 26 | 22 | 16 |
| 1100101 | 145 | 101 | 65 |
| 10110011 | 263 | 179 | B3 |