Practice Exam CIS 351

Question 1 Warm up (*20 points*) – For each question, write on the space to the left the answer that best matches the answer to the question or best answer to complete the question.

____1. If a loop does not contain within itself a way to terminate, it is called a(n)

a. `while` loop
b. `do-while` loop
c. `for` loop
d. infinite loop

____2. This type of loop will always be executed at least once.

a. pre-condition loop
b. post-condition loop
c. sentinel loop
d. for loop

____3. What will be the value of `x` after the following code is executed?

```
int x;
x = 0;
while (x <= 100)
{
   x = x + 10;
}
```

a. 90
b. 100
c. 110
d. This is an infinite loop

____4. What will be the value of `x` after the following code is executed?

```
int x;
int y;
x = 10;
y = 20;
while (y <= 100)
{
   x = x + y;
}
```

a. 90
b. 110
c. 210
d. **This is an infinite loop**

____5. In the following code, what values could be read into number to terminate the while loop?

```
Scanner keyboard;
int number;
keyboard = new Scanner(System.in);
System.out.print("Enter a number:  ");
number = keyboard.nextInt();
while (number < 0 || number > 100)
{
   System.out.print("Enter another number:  ");
   number = keyboard.nextInt();
}
```

a. Any number less than 0 or greater than 100
b. **Any number in the range 0 – 100**
c. Any number in the range -1 - 101
d. The `boolean` condition can never be true

____6. This is a value that signals when the end of a list of values has been reached.
   a. Terminal value
   b. Final value
   c. End value
   d. **Sentinel value**

____7. Before entering a loop to compute a running total, the program should first do this.
   a. **Read all the values into main memory**
   b. **Set the accumulator where the total will be kept to an initial value, usually zero**
   c. **Know exactly how many values there are to total**
   d. Set all variables to zero

____8. In all but rare cases, loops must contain within themselves
   a. arithmetic statements
   b. `if` statements
   c. **a way to terminate**
   d. nested loops

____9. Methods are commonly used to
   a. speed up the compilation of a program
   b. break a problem down into small manageable pieces
   c. emphasize certain parts of the logic
   d. document the program

____10. Which of the following are pre-test (pre-condition) loops?
   a. `while`, `for`, `do-while`
   b. `while`, `do-while`
   c. `while`, `for`
   d. `for`, `do-while`

____11. If `chr` is a character variable, which of the following `if` statements is written correctly?
   a. `if (chr == 'a')`
   b. `if (chr.equals('a'))`
   c. `if (chr = "a")`
   d. `if (chr.equals("a"))`
   e. They are all written correctly.

____12. What would be the value of `discountRate` after the following statements are executed?

```
double discountRate;
int purchase;
purchase = 100;
if (purchase > 1000)
   discountRate = .05;
else if (purchase > 500)
   discountRate = .03;
else if (purchase > 100)
   discountRate = .01;
else
   discountRate = .00;
```

   a. .05
   b. .03
   c. .01
   d. 0.0

____13. If `str1` and `str2` are both `String`s, which of the following will correctly test to determine whether `str1` is less than `str2`?

(1) `(str1 < str2)`
(2) `(str1.equals(str2) < 0)`
(3) `(str1.compareTo(str2) < 0)`

a. 1, 2, and 3 will all work
b. 2 and 3
c. 2, only
d. **3, only**

____14. Enclosing a group of statements inside a set of braces creates a
a. **block of statements**
b. `boolean` expression
c. loop
d. Nothing, it is just for readability

____15. (**T**/F) A local variable's scope always ends at the closing brace of the block of code in which it is declared.

____16. (T/**F**) When testing for character values, the `switch` statement does not test for the case (upper or lower) of the character.

____17. (**T**/F) Constants, variables, and the values of expressions may all be passed as arguments to a method.

____18. (**T**/F) A parameter variable's scope is the method in which the parameter is declared.

____19. If method A calls method B, and method B calls method C, and method C calls method D, when method D finishes, what happens?
a. control is returned to method A
b. control is returned to method B
c. **control is returned to method C**
d. the program terminates

____20. When an argument is passed to a method,
a. **its value is copied into the method's parameter variable**
b. its value may be changed within the called method
c. values may not be passed to methods
d. the method must not assign another value to the parameter that receives the argument

Question 2 Tracing (*6 points*) – For each question, list on the space to the left the answer that best matches the question. **Show work for partial credit.**

_____1. What will be the value of `x` after the following code is executed?

```
int x;
x = 10;
for (int y = 8; y < 23; y = y + 5)
   x = x + y;
```

a. 31
b. 49
c. 72
d. Invalid `for` statement

_____2. What will be the value of `tickets` after the following code is executed?

```
int persons;
int age;
double tickets;
persons = 5;
age = 50;
if (persons == 1)
   if (age < 50)
      tickets = 12.50;
   else
      tickets = 10.00;
else if (persons > 1 && persons < 5)
   if (age <= 50)
      tickets = persons * 12.00;
   else
      tickets = persons * 9.50;
else
   tickets = persons * 9.00;
```

a. 45.00
b. 47.50
c. 50.00
d. 60.00
e. 62.50

Question 3 (*6 points*) Short coding– Some of the questions below ask you to write a statement, others an expression. You must answer with the entity the question asks for. **DO NOT** write a declaration statement unless the question explicitly asks you to do so.

THESE ANSWERS ARE PRETTY BASIC, so COMPLETE ON YOUR OWN

a. Write a Java expression that returns `true` if the variable `name` contains the `String` "`Smith`" and `false` otherwise.

b. Write a Java **expression** that evaluates to `true` if an `int` number, `myVal`, is evenly divisible by 3 and 5 and `false` otherwise.

c. Write a Java **if statement** that prints "`Great!`" only if the `double` variable `score` is greater than `90`.

Question 4 *(10 points)* (Expression Evaluation) - Fill in the chart below with the result of each of the **Java** expressions. Write INVALID in both boxes if there is an invalid expression (one that will not compile). All decimal calculations should be carried out to 2 places only. (You should NOT use a calculator for this problem). Assume all variables have been declared and initialized.

ANSWER OF A SIMILAR QUESTIONS IS SOLVED DURING DECISION LECTURE

Expression	Data Type of result	Result
`(double)  (3 / 6)`		
`(!(a == a)) || (b == b)`		
`true || ( (17 / 4) < (72 - 68))`		
`(17 - 5) < 15 < (25 + -9)`		
`(14 < (28 - 7)) && ((24 * 3) < (8 * 9))`		

Question 5 *(10 points)* (Problem solving) – Choose **one** of the two problems below to solve.

a) **THINK BEFORE YOU WRITE.** Write a method, `makeSquare`, that takes in a single `int` parameter, `size`, and produces a square on the screen using the letters `X` and `O`. The number of characters in each line and the number of rows printed will be `size`. The particular letter is determined by the row and column. If the row and column numbers match, print an `X`; otherwise print an `O`. For example if the `size` parameter is `4`, your square would look like:

XOOO
OXOO
OOXO
OOOX

If a zero or negative number is entered, print nothing.

```
public static void makeSquare(int size)
{
for (int i = 0; i < 4; i++) {
 6          for (int j = 0; j < 4; j++) {
 7              if (i == j)
 8                 System.out.print("X");
 9              else
10                 System.out.print("O");
11          }
12          System.out.println();
13      }
```

}

b) **THINK BEFORE YOU WRITE.** A picture frame manufacturer wants a method that will print all of the different sizes of frames that they make. The method should take in the smallest dimension and the largest dimension and should print a chart of each combination of sizes in increments of 2 (from the smallest to the largest). So if the smallest frame were 1 inch and the largest 5 the chart would look like:

1 x 1 1 x 3 1 x 5
3 x 1 3 x 3 3 x 5
5 x 1 5 x 3 5 x 5

If the largest is less than or equal to the smallest or either of the dimensions is less than or equal to 0, print nothing.

```
public static void makeChart(int small, int large)
{


for (int i = 1; i <= 5 ; i+=2) {
 6          for (int j = 1; j <= 5; j+=2) {
 7
 8                 System.out.print(i + "X" + j);
 9                 System.out.print("\t");
10          }
11          System.out.println();
12      }
```

}

c) What would be the results after the following code was executed?

```
int[] x = { 23, 55, 83, 19 };
int[] y = { 36, 78, 12, 24 };
x = y;
y = x;
```

a.	x[] = { 36, 78, 12, 24 } and y[] = { 23, 55, 83, 19 }
b.	**x[] = { 36, 78, 12, 24 } and y[] = { 36, 78, 12, 24 }**
c.	x[] = { 23, 55, 83, 19 } and y[] = { 23, 55, 83, 19 }
d.	This will produce a compilation error.

d) Given the following java program:

```
public class ActivityGenerator
{
   public static void main(String [] args)
   {
      int number;

      String[] activities = {"Sleep", "Play Basketball",
         "Shop", "Take a drive"};

      if (args.length > 0)
      {
         number = Integer.parseInt(args[0]);
         if (number < 0)
            number = 0;
      }
      else
         number = 1;

      System.out.println(activities[number % activities.length]);

   }
```

```
}
```

What is output by the follow program calls from the command line?

a) java ActivityGenerator

```
Play Basketball
```

b)java ActivityGenerator 3

```
"Take a drive"
```

c)java ActivityGenerator 13

```
Play Basketball
```

Question 5 – (24 points) Tracing

Using BankAccount.java and AccountTestV2.java, trace the code. At each break point, indicate the value of each of the variables listed. Use the provided grid paper or back of another page for keeping track of your values. You may also choose to draw a model or diagram of the various objects as you work with them.

Part A: Execute AccountTestV2 through line A-18
A1. How many "containers" exist in memory for the following BankAccount attributes?

RATE_____1_____ nextAccount _____1_____ number_____2_____

A2. How many BankAccount objects currently exist? _____2_____

A3. What is is output by lines **A-15** and **A-16**

```
Account 1234    Balance: $50.00
Account 1235    Balance: $100.00
```

A4. What is the value of BankAccount.allAccounts? _150_____

Part B: Continue to execute AccountTestV2 through line A-24
B1. What is the value of nancy.balance?___0_____ vinny.balance?_____-150_____

B2. What is the value of BankAccount.allAccounts?___100_____

Part C: Continue to execute AccountTestV2 through line A-37

C1. What is the value of cs139.length? _5_____

C2. What are the values of: cs139[0].balance? ___10_____
cs139[1].balance _____50_____
cs139[2].balance _____10__
cs139[3].balance _____200_____
cs139[4].balance _____0__

C3. What is the value of BankAccount.allAccounts? _____262_____

C4. How many BankAccount objects exist in memory? _____5_____

Part D: Continue to execute AccountTestV2 through line A48

D1. What is output by lines **A-42** and **A-46**

```
Account 1234    Balance: $10.20
Account 1235    Balance: $50.50
Account 1234    Balance: $10.20
Account 1236    Balance: $202.00
Account 1237    Balance: $0.00
```

Question 6 – Writing Program (10 pt)

(10 points) **Part 9 – Solve a problem** – Given the following main method, write the code to carry out the specified task.

Your program should read a double value representing the price of a purchased item from the keyboard, calculate the amount of tax on the item and then display the item amount, tax, and total price in a line of the form:

_____ cost + _____ tax = _____. , where the blanks are filled by the item amount, tax and total price respectively. The tax rate is a fixed 5%. NOTE: a bit of code has been provided for you.

```
public static void main (String [] args)
{  // declare your variables and/or constants here
   final double TAX_RATE = .05;
   double amount;
   double tax;
   double totalPrice;
   Scanner keyboard;
   keyboard = new Scanner(System.in);
   // prompt for the input and obtain the input.
   System.out.print("Please enter the item amount: ");
   amount = keyboard.nextDouble();
   // calculation and output
   tax = amount * TAX_RATE;
   totalPrice = amount + tax;
   System.out.println(amount + " cost + " + tax + " tax = " + totalPrice + ".");
}
```

Some helpful Scanner methods:

nextLine(): returns an entire line up to and including the new line character

next(): returns the next token from a line

nextInt(): returns the next integer from a line

nextDouble(): returns the next double value from a line

Question 7 – Writing Program (20 pt)

```
 1 import java.util.*;
 2 /** a class to represents standard time measures
 3  *
 4  * @author - Zamua Nasrawt
 5  * @version - 1.0 - 12/9/2014
 6  */
 7     public class Time139
 8    {
 9       private int hour;     // hours - must be greater than 0
10       private int minute;   // minutes - must be between 0 and 59
11
12    /** Default constructor, sets the time to 0 hours and 0 minutes
13    */
14       public Time139 ()
15      {
16         this.hour = 0;
17         this.minute = 0;
18      }
19
20    /** Explicit value constructor.
21    *  If the incoming hour OR incoming minute are less than 0,
22    *  set both hour and minute to zero.
23    *  If the seconds are greater than 59, adjust the corresponding
24    *  hours and minutes.
25    *
26    *  1 hour = 60 minutes
27    *
28    *  @param hour - number of incoming hours
29    *  @param minute - number of incoming minutes
30    */
31       public Time139 (int hour, int minute)
32      {
33         if(hour < 0)
34         {
35            this.hour = 0;
36         }
37
38         if(minute < 0)
39         {
```

```
40             this.minute = 0;
41          }
42
43          if(minute > 59)
44          {
45             this.hour += (minute/60);
46             this.hour += (minute % 60);
47          }
48       }
49
50     /** adjustTime is a private helper method that adjusts the time
51      *  attributes so that minute attribute is between 0 and 59.
52      *  If the minutes exceeds 59, adjust the hours and minutes based
on
53      *  1 hour = 60 minutes.
54      */
55        private void adjustTime()
56       {
57          if(minute > 59)
58          {
59             this.hour += (minute/60);
60             this.hour += (minute % 60);
61          }
62       }
63
64      /** setTime sets this Time139 object to the value of the Time139
65      *  object passed in as newTime.
66      *
67      * @param newTime - The Time139 object to set this time object
values
68      *                  to
69      */
70        public void setTime(Time139 newTime)
71       {
72          this.hour = newTime.hour;
73          this.minute = newTime.minute;
74       }
75
76      /** compareTo compares this time to the other time and returns
77      *  -1 if this time is less than other time, 0 if they are the
78      *  same and 1 if this time is greater than other time.
79      *  They are considered the same if all of their fields match.
80      *
81      * @param other The time to compare
82      * @return -1 if this is less than other, 0 if they are the same
83      *         1 if this time is greater than other
84      */
85       public int compareTo(Time139 other)
86       {
87          if(this.hour < other.hour)
88          {
89             return -1;
```

```
 90          }
 91
 92          if(this.hour > other.hour)
 93          {
 94             return 1;
 95          }
 96
 97          if(this.hour == other.hour)
 98          {
 99             if(this.minute < other.minute)
100             {
101                return -1;
102             }
103
104             if(this.minute > other.minute)
105             {
106                return 1;
107             }
108          }
109
110          return 0;
111       }
112
113     /** creates a new Time139 object which is a duplicate of this time
114     *
115     * @return a Time139 object which contains the same values
116     *  as this time object.
117     */
118        public Time139 duplicate ()
119       {
120          Time139 klone;
121
122          klone = new Time139();
123
124          klone.hour = this.hour;
125          klone.hour = this.minute;
126
127          return klone;
128       }
129
130      /** This method returns a valid positive integer.
131      *  The method should continue to prompt the user for a value
until an
132      *  acceptable value is entered.
133      *
134      *
135      *  @param prompt A prompt message
136      *  @return A valid positive integer number
137      */
138      public static int getNumber(String prompt)
139      {
140          Scanner scn;
```

```
141          int val1 = 0;
142          int val2 = 0;
143
144          scn = new Scanner(System.in);
145
146          System.out.println(prompt);
147
148          while(!scn.hasNextInt())
149          {
150             System.out.println(prompt);
151             scn.next();
152
153             if(scn.hasNextInt())
154             {
155                if(val1 <= 0)
156                {
157                   System.out.println(prompt);
158                   scn.next();
159
160                }
161                if(val1 >= 0)
162                {
163                   val2 = val1;
164                }
165             }
166          }
167
168          return val2;
169       }
170
171
172    /** toString returns a String representation of this Time139 object
173     *  It is provided here to use in testing if you wish
174     *
175     * @return A String representing this Time139
176     */
177       public String toString()
178       {
179          return hour + " hours and " + minute + " minutes";
180       }
181
182    /** min
183     *
184     *  This method takes in an array of doubles and returns
185     *  a value that is the minimum value of all the elements of
186     *  the array.  You must take into account the fact that the
187     *  array may be empty.  Return Double.MIN_VAL if the array has
188     *  no elements or is null. If the array were {4, 5, 22, 1.5, 7.9}
189     *  the method should return 1.5
190     *
191     *  @param arr1 - The array to search
192     *  @return The minimum of the array elements
```

```
193     */
194        public static double min(double [] arr1)
195       {
196          double min;
197          double minRecord;
198
199          if(arr1.length == 0 || arr1 == null)
200          {
201             return Double.MIN_VAL;
202          }
203
204          min = arr1[0];
205
206          for(int i = 1; i < arr1.length; i++)
207          {
208             if(arr1[i] < arr1[i-1])
209             {
210                minRecord = arr1[i];
211             }
212          }
213
214          if(minRecord < min)
215          {
216             min = minRecord;
217          }
218
219          return min;
220       }
221    }
222
```