Name: ___  KEY

<table>
<thead>
<tr>
<th>Question</th>
<th>Value</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>100</strong></td>
<td></td>
</tr>
</tbody>
</table>

Please answer questions in the spaces provided. Please be legible. If you make a mistake or need more space, use backs of pages - clearly indicate where the answer can be found.

*Good luck and best wishes for a great summer!*
1. (________/10) What gets printed? Please show output as it will appear, or indicate "NO OUTPUT". If there is an infinite loop, be sure to show some lines of the output followed by "... INFINITE LOOP".

```java
int a = 4;
do {
    a--;
    System.out.println(a);
} while (a < 4)
```

**Output:**
```
3
2
1
... INFINITE LOOP
```

```java
String[] word = {"up", "down"};
for (String w1 : word)
    for (String w2 : word)
        System.out.println("yes " + w1 + " " + w2 + " " + w1);
```

**Output:**
```
yes up up up
yes up down up
yes down up down
yes down down down
```

```java
int size = 10;
do {
    System.out.print(size + " => ");
    int category = size / 5;
    switch(category) {
        case 0:
            System.out.println ("S");
            break;
        case 1:
            System.out.println ("M");
            break;
        default:
            System.out.println ("L");
    }
    size = size - 2;
} while (size > 2);
```

**Output:**
```
10 => L
8 => M
6 => M
4 => S
```

```java
boolean flipFlop = false;
for (int i = 3; i>=1; i--)
    {
        flipFlop = !flipFlop;
        System.out.println(i + " " + flipFlop);
    }
```

**Output:**
```
3 true
2 false
1 true
```
2. (___/10)

a. Complete the code for the applet that produces the image to the left.
import javax.swing.JApplet;
import java.awt.*;
public class Mystery extends JApplet {
    public void paint (Graphics page) {
        page.drawLine (10, 60, 90, 60);
        page.drawLine (20, 20, 40, 80);
        page.drawRect (0, 40, 60, 40);
        page.fillOval (40, 0, 80, 40);
    }
}

b. Given a Random object named gen, what range of values are produced by the following expressions?
• gen.nextInt(8) __________ 0 to 7
• gen.nextInt(30) + 100 __________ 100 to 129
• gen.nextInt(3) – 2 __________ -2 to 0

c. Suppose you have the following declarations; evaluate some expressions and show the output produced:

String word = "kumbaya";
int num = word.length();

int n = 0;
while (n < word.length())
{
    System.out.print("*");
    n = n + 2
}

Output:
*km*a*a
3. (_____/10)

a. Write a Java method `birthday()` with one `int` parameter `n` that prints “Happy Birthday” `n` times. The method should not return anything. For example, if the method is invoked like this: `birthday(3)`, it should print the following:

```
Happy Birthday
Happy Birthday
Happy Birthday
```

```java
void birthday(int n) {
    for (int i = 0; i < n; i++)
        System.out.println("Happy Birthday");
}
```

b. Write a Java method `stars()` with one `int` parameter `n` that prints the shape below. The method should not return anything. For example, if the method is invoked like this: `stars(5)`, it should print the following:

```
* 
**
***
****
*****
```

```java
void stars(int n) {
    for (int i = 1; i <= n; i++)
        for (int j = 0; j < i; j++)
            System.out.print("*");
        System.out.println();
}
```
Answer Key

4. (_____/ 10) Consider the following program:

```java
public class ArrayDemo {
    public static void main(String[] args) {
        int[] anArray; // declares an array of integers
        anArray = new int[10]; // allocates memory for 10 integers
        anArray[0] = 100; // initialize first element
        anArray[1] = 200; // initialize second element
        anArray[2] = 300; // etc.
        anArray[3] = 400;
        anArray[4] = 500;
        anArray[5] = 600;
        anArray[6] = 700;
        anArray[7] = 800;
        anArray[8] = 900;
        anArray[9] = 1000;
        System.out.println("Element at index 0: " + anArray[0]);
        System.out.println("Element at index 1: " + anArray[1]);
        System.out.println("Element at index 2: " + anArray[2]);
        System.out.println("Element at index 3: " + anArray[3]);
        System.out.println("Element at index 4: " + anArray[4]);
        System.out.println("Element at index 5: " + anArray[5]);
        System.out.println("Element at index 6: " + anArray[6]);
        System.out.println("Element at index 7: " + anArray[7]);
        System.out.println("Element at index 8: " + anArray[8]);
        System.out.println("Element at index 9: " + anArray[9]);
    }
}
```

- Sketch the contents of `anArray` (be sure to include indices in your diagram).

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>200</td>
<td>300</td>
<td>400</td>
<td>500</td>
<td>600</td>
<td>700</td>
<td>800</td>
<td>900</td>
<td>1000</td>
</tr>
</tbody>
</table>

- What happens if the `anArray` is not initialized (if we comment out the code in (a)).
  all the values are set to zero
- Rewrite a: Write a for-loop to replace the repetitive code in (a), so that the array is initialized to exactly the same values as before.

```java
for (int i = 0; i < anArray.length; i++)
    anArray[i] = i*100 + 100;
```
- Rewrite (b): Write a for-loop to replace the repetitive code in (b), so that the program produces exactly the same output as before.

```java
for (int i = 0; i < anArray.length; i++)
    System.out.println("Element at index " + "i" + ": " + anArray[i]);
```
5. (——/10)

Draw diagrams showing the contents of array \texttt{table} after execution of the following code fragments. (Be sure to include indices in your diagram).

```java
int[][] table = new int[2][3];
for (int i=0; i < 2; i++)
    table[i][2] = i + 1;
```

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
```

```java
char[][] table = new char[4][2];
String sample = "Alternative facts.";
int count = 0;
for (int i=0; i < 4; i++)
    for (int j=0; j < 2; j++)
    {
        table[i][j] = sample.charAt(count);
        count++;
    }
```

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>'A'</td>
<td>'l'</td>
</tr>
<tr>
<td>1</td>
<td>'t'</td>
<td>'e'</td>
</tr>
<tr>
<td>2</td>
<td>'r'</td>
<td>'n'</td>
</tr>
<tr>
<td>3</td>
<td>'a'</td>
<td>'t'</td>
</tr>
</tbody>
</table>
```

```java
int[][] table = new int[3][4];
for (int i=0; i < table.length; i++)
    for (int j=0; j < table[i].length; j++)
        table[i][j] = i / (j+1);
```

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```
a. Trace through the following code and show what gets printed.

```java
int[] a = {40, 80, 120, 160};
int[] b = {1000, 2000, 3000, 4000};
int[] c = b;

for (int i=0; i<a.length; i++)
    a[i] = b[i];
a[1] = 25;
b[2] = 50;
c[3] = 77;
for (int x: a)
    System.out.print(x + " ");
System.out.println();
for (int x: b)
    System.out.print(x + " ");
System.out.println();
for (int x: c)
    System.out.print(x + " ");
System.out.println();
```

**Output:**

```
1000 25 3000 4000
1000 2000 50 77
1000 2000 50 77
```

b. Suppose an int array ratings contains values in the range 0-3. Write a code fragment that creates an array count that contains the frequency of occurrences of 0’s, 1’s, 2’s and 3’s in the ratings array, and then prints this information.

**Example:** if the array ratings has these contents:

```
ratings
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
```

Your code should create the array count with the following contents:

```
count
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
```

and the output would be:

Count for 0: 1
Count for 1: 1
Count for 2: 4
Count for 3: 1

Write your code fragment below. Assume the array ratings is already initialized to some values. Your code should work for any size array.

```java
int[] count = new int[4];
for (int x: ratings)
    count[x]++;
for (int i = 0; i < count.length; i++)
    System.out.println ("Count for " + i + ": " + count[i]);
```
Answer Key

7. (___/10)
    a. Write a method named `randomYak` with one parameter, an array of `String`, that returns a randomly selected `String` from the array.

```java
String randomYak(String[] a)
{
    Random rand = new Random();
    return a[rand.nextInt(a.length)];
}
```

b. Write a method named `shuffle` with one parameter, an array of `String`, that shuffles the elements of the array.

```java
void shuffle(String[] a)
{
    Random rand = new Random();

    // shuffling enough times (not specified in question)
    // using 2 * a.length
    for (int i = 0; i < 2 * a.length; i++)
    {
        int x = rand.nextInt(a.length);
        int y = rand.nextInt(a.length);
        String temp = a[x];
        a[x] = a[y];
        a[y] = temp;
    }
}
import java.text.NumberFormat;

public class Billing {
    public static void main (String[] args) {
        Person[] subject;
        subject = new Person[3];

        subject[0] = new Person("Mr Bean");
        subject[1] = new Person("person 1");
        subject[2] = new Person("person 2");

        subject[0].bill(130);
        subject[1].bill(215);
        subject[2].bill(85);

        System.out.println(subject[0]);
        System.out.println(subject[1]);
        System.out.println(subject[2]);

        subject[2].bill(60);

        for (int i = 0; i < subject.length; i++)
            System.out.println(subject[i]);

        int sum = 0;
        for (int i = 0; i < subject.length; i++)
            sum = sum + subject[i].getBalance();

        NumberFormat fmt = NumberFormat.getCurrencyInstance();

        System.out.println("Total balance:" +
            fmt.format(sum));
    }
}
8. (____/10) Suppose you have a Person class defined as follows:

```java
import java.text.NumberFormat;
public class Person
{
    private String name;
    private double balance;

    public Person (String x) // constructor
    {
        name = x;
        balance = 0;
    }

    public double getBalance() // balance accessor
    {
        return balance;
    }

    public void bill(double amount) // bills a Person
    {
        // adds to balance
        balance = balance + amount;
    }

    public String toString () // toString()
    {
        NumberFormat fmt = NumberFormat.getCurrencyInstance();
        return (name + "\t" + fmt.format(balance));
    }
}
```

On the facing page, write the Java code for a client that implements the following algorithm:
- Declare (but do not instantiate) an array named subject that stores Person objects
- Instantiate an array of three Person objects and assign it to subject
- Create a Person object for someone named “Mr Bean”, and store it as the first element in the array subject
- Create two more Person objects and store them as the next two elements in the array subject
- Use the bill() method to add $130, $215, and $85 to the balances of the Person objects stored in the array subject.
- Print the information about all the Person objects stored in the array subject. (do NOT use a loop, use repetitive code).
- Add $60 to the bill of the last Person object stored in the array subject.
- Print the information about all the Person objects stored in the array subject. (this time use a for loop).
- Compute and print the total balance of all the subjects.
**Answer Key**

9. (_____ / 10) Consider the program `ResponseTimeExperiment.java` on the following page.

a. Rewrite the code in the box labeled (a) using the conditional operator instead of if/else.

```java
int outcome = (number == a + b) ? 1: 0;
```

b. Rewrite the code in the box labeled (b) using the conditional operator instead of if/else.

```java
System.out.println(outcome == 1? "Correct!" : "Incorrect.");
```

c. Annotate the code on the next page with the number to match each step in the algorithm.

d. Rewrite the algorithm, modifying it to:
   - Repeat the experiment 4 times
   - Count the number of correct responses
   - Compute and print the average response time.

   DO NOT write java code, just rewrite the algorithm. (Use facing page if you need more room for your answer.)

   **Algorithm:**
   1. input name
   2. print personalized welcome message & instructions
   3. wait for user to hit "ENTER"
   4. a = random int
   5. b = random int
   6. startTime = current time
   7. print question using a, b
   8. input answer
   9. endTime = current time
   10. outcome = 1 or 0 (answer is correct or incorrect)
   11. reactionTime = endTime - startTime
   12. print outcome as "Correct" or "Incorrect"
   13. print reaction time
   14. print "goodbye"

   **Algorithm:**
   1. input name
   2. print personalized welcome message & instructions
   3. wait for user to hit "ENTER"
   4. a = random int
   5. b = random int
   6. startTime = current time
   7. print question using a, b
   8. input answer
   9. endTime = current time
   10. outcome = 1 or 0 (answer is correct or incorrect)
   11. reactionTime = endTime - startTime
   12. print outcome as "Correct" or "Incorrect"
   13. print reaction time
   14. print "goodbye"

1. input name
2. print personalized welcome message & instructions
   A. numCorrect = 0
   B. totalTime = 0
   C. repeat 4 times:
      3. wait for user to hit "ENTER"
      4. a = random int
      5. b = random int
      6. startTime = current time
      7. print question using a, b
      8. input answer
      9. endTime = current time
      10. outcome = 1 or 0 (answer is correct or incorrect)
      11. reactionTime = endTime - startTime
      12. print outcome as “Correct” or “Incorrect”
   D. numCorrect = numCorrect + outcome
   E. totalTime = totalTime + reactionTime
   F. averageTime = totalTime/4
   G. print averageTime
   13. print goodbye
// ResponseTimeExperiment.java
// Measure response time for addition problems.
// M A Papalaskari

import java.util.Scanner;
import java.util.Random;

public class ResponseTimeExperiment {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        Random rand = new Random();

        System.out.print("Please enter your name: ");
        String name = in.nextLine(); 1

        System.out.println("Hello " + name + " Please answer as fast as you can.");
        System.out.println("\n\nHit <ENTER> when ready.");
        in.nextLine(); // wait for user to hit <ENTER> 3

        int a = rand.nextInt(100); 4
        int b = rand.nextInt(100); 5

        long startTime = System.currentTimeMillis(); 6

        System.out.print(a + " + " + b + " = ");
        String response = in.nextLine(); 7
        int number = Integer.parseInt(response);
        long endTime = System.currentTimeMillis(); 9

        int outcome;
        if (number == a + b) a
            outcome = 1;
        else
            outcome = 0;

        long reactionTime = endTime - startTime; 11

        if (outcome == 1) b
            System.out.println("Correct!");
        else
            System.out.println("Incorrect.");

        System.out.println("Time: " + reactionTime + "milliseconds"); 13

        System.out.println("Thank you " + name + ", goodbye."); 14
    }
}
10. (_____ / 10) Consider the program ResponseTimeExperiment.java. The code is reproduced on the next page to make it easier for you to answer this question. This question is independent of the previous question (does NOT assume you have answered the previous question).

a. Suppose you wish to create a text file, named name.txt, where name is the String entered by the user when prompted to enter their name. Annotate the program on the next page to show where to insert the following code fragments to accomplish this:
   1.(outfile.println((number == a + b) + "\t" + reactionTime);
   2. import java.io.*;
   3. throws IOException
   4. outfile.close();
   5. PrintWriter outfile = new PrintWriter (name + ".txt"));

b. Which of the java code fragments above might throw an IOException? (note the corresponding number) ___5_____

c. Suppose you wish to prevent files with certain names from being created. You have a list of banned words saved in a file named banned.txt and if the user enters any of these words as their name, the program should throw an exception and terminate. Assume the class BadWordException is already defined. Show where would you place the following code in ResponseTimeExperiment.java

   Scanner badWords = new Scanner(new File("banned.txt"));
   while (badWords.hasNext())
      if (name.equals(badWords.next()))
         throw (new BadWordException("Watch your tongue"));

d. Suppose you wish to handle NumberFormatException exceptions using the following catch clause. Draw a rectangle around the code on the next page, indicating where to place the try block.

   catch (NumberFormatException problem)
   {
      System.out.println("FAIL: You were supposed to type in a number");
      outfile.println("*");
   }
// ResponseTimeExperiment.java
// Measure response time for addition problems.
// M A Papalaskari

import java.util.Scanner;
import java.util.Random;

public class ResponseTimeExperiment
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        Random rand = new Random();

        System.out.print("Please enter your name: ");
        String name = in.nextLine();

        System.out.println("Hello "+ name + ". Please answer as fast as you can.");
        System.out.println("\n\nHit <ENTER> when ready.");

        in.nextLine(); // wait for user to hit <ENTER>

        int a = rand.nextInt(100);
        int b = rand.nextInt(100);

        long startTime = System.currentTimeMillis();
        System.out.print(a + " + " + b + " = ");
        String response = in.nextLine();

        int number = Integer.parseInt(response);

        long endTime = System.currentTimeMillis();
        if (number == a + b)
            outcome = 1;
        else
            outcome = 0;

        long reactionTime = endTime - startTime;

        if (outcome == 1)
            System.out.println("Correct!");
        else
            System.out.println("Incorrect.");

        System.out.println("Time: "+ reactionTime + "milliseconds");

        System.out.println("Thank you "+ name + ", goodbye.");
    }
}
Boolean Expressions

Booleans are used in Java to represent values that must be true or false. They use boolean expressions to determine what to do next.

Example:

```java
if (credits == 0)
    System.out.println("GPA: None"); // true do this
else
    if (gpa <= qP / credits)
        System.out.println("\n\n\nGPA: " + gpa); // false do this
```

The Conditional Operator Syntax

```
condition ? expression1 : expression2
```

- If the `condition` is true, `expression1` is evaluated; if it is false, `expression2` is evaluated.
- The value of the entire conditional operator is the value of the selected expression.
- Example: Rewrite this using conditional operator:
  ```java
  if (n > 0)
      System.out.print("positive");
  else
      System.out.print("negative");
  ```

The Conditional Operator Syntax

```
switch (expression)
```

- `expression` can be an integer, char, or enumerated type.
- `case` values can be integers, floating point values, NO ranges of values (e.g. 0 <= x < 10).

- If expression matches `value1`, control jumps to `statement-list1`.
- If expression matches `value2`, control jumps to `statement-list2`.
- If expression matches `value3`, control jumps to `statement-list3`.
- If none of the values match the expression, control jumps to here.

- If expression matches `value1`, control jumps to `statement-list1`.
- If expression matches `value2`, control jumps to `statement-list2`.
- If expression matches `value3`, control jumps to `statement-list3`.
- If none of the values match the expression, control jumps to here.

- If expression matches `value1`, control jumps to `statement-list1`.
- If expression matches `value2`, control jumps to `statement-list2`.
- If expression matches `value3`, control jumps to `statement-list3`.
- If none of the values match the expression, control jumps to here.

- If expression matches `value1`, control jumps to `statement-list1`.
- If expression matches `value2`, control jumps to `statement-list2`.
- If expression matches `value3`, control jumps to `statement-list3`.
- If none of the values match the expression, control jumps to here.

The Conditional Operator Syntax

```
while (condition)
```

- If `condition` is true, `statement` is executed.
- Control continues to `while` if `condition` is true.
- The body of a while loop executes at least once.

```
do {
    statement
    increment
} while (condition);
```

- Increment is done after `statement` is executed.
- Loop continues as long as `condition` is true.
- The body of a do loop executes at least once.

```
for (int count = 0; count < 5; count++)
```

- `count` is initialized to 0.
- `count` is incremented at the end of each iteration.
- Control continues to `for` as long as `count` is less than 5.
- The body of a for loop executes at least once.
**Example: Stars.java**

```java
public class Stars {
    public static void main(String[] args) {
        final int MAX_REDS = 10;
        for (int row = 1; row <= MAX_REDS; row++) {
            for (int star = 1; star <= row; star++)
                System.out.print("*");
            System.out.println();
        }
    }
}
```

**UML Class Diagrams**

**UML** = Unified Modelling Language

- Example: A UML class diagram for the RollingDice program:

```
public class RollingDice {
    private int faceValue;
    public int getFaceValue() {
        return faceValue;
    }
    public void setFaceValue(int value) {
        faceValue = value;
    }
    public String toString() {
        return faceValue + " \" faceValue \"";
    }
}
```

**Arrays**

- Declaration: `double[] scores = new double[10];`

- Initialization: `scores = [7.9, 8.7, 9.4, 8.2, 9.8, 8.7, 8.1, 7.4, 9.1];`

- Size of array: `scores.length = 10`

**Arrays as Parameters**

- Example:

```java
int[] ratings = {4, 3, 3, 1, 4, 3, 1, 0, 3, 4};
System.out.println(average(ratings));

public static double average(int[] a) {
    for (int num : a)
        sum += num;
    return (double) sum / a.length;
}
```

**2D Arrays - Overview**

```
declaration double[][] courseGrade = new double[3][10];
```
Zero.java  -- updated

try / catch
- Create a try block surrounding code that we think may cause an exception
- catch clause has code to tell it what to do
- The exception handler
  - Can have multiple catch clauses
  - One for each type of exception thrown by try block
- If no exception is thrown, processing continues following the try statement (skips catch clauses)

But is the trace gone???
- No
- Methods exist to get the trace and system error
  - getMessage()  Returns a string explaining the reason the exception was thrown
  - printStackTrace()  Prints the call stack trace indicating where the error occurred

Unchecked Exceptions

```java
public class Zero {
    public static void main(String[] args) {
        int numerator = 10;
        int denominator = 0;
        try {
            System.out.println(numerator / denominator);
        } catch (ArithmeticException e) {
            System.out.println("Bad division");
        } catch (Exception e) {
            System.out.println("This will not print");
        }
    }
}
```