Instructions:

KEEP TEST BOOKLET CLOSED UNTIL YOU ARE INSTRUCTED TO BEGIN.

Omit one page from this exam. To select the page, cross out the entire page in one large 'X'. If you do not select a question for omission, the last question of the exam will be omitted for you.

This exam is double sided (front and back)!

No outside resources (calculators, notes, textbooks, etc.) are allowed except writing utensils. If you need extra scratch paper, please pick it up from the front of the class.

All answers must be submitted on (or attached to) this exam. All answers must be clearly legible.
True/False (10 points):
Instructions: Read the entire statement. Circle whether this statement is either True or False.

1. A “byte” is the most basic logical unit that can be stored in a computer and it stores the value of either a 1 or a 0.
   true  false

2. Every java program has to have a method defined as 'public static void main(String[] args)' in order to run.
   true  false

3. The type of a specific variable may change over the course of a program.
   true  false

4. Java bytecode is an intermediary step between source code and machine code. As a result bytecode is tied to the hardware that the program is compiled on.
   true  false

5. What is the output of the following expression:

   System.out.println(false && ((false || false) && (true) || (true && true)));

   true  false
Multiple Choice (10 points):
Instructions: Circle the best answer(s).

1. Suppose “isPrime” is a boolean variable, which of the following are valid expressions? (circle all that apply)
   A. if (isPrime != true)
   B. if (isPrime = true)
   C. if (isPrime)
   D. if (isPrime == false)
   E. if (!isPrime = false)

2. The JDK (Java Development Kit) includes which of the following components: (circle all that apply)
   A. Java browser
   B. Java interpreter
   C. Java compiler
   D. Java runtime libraries

3. The hexadecimal value of the number 15 is:
   A. 0xF
   B. 0xA
   C. 0x11
   D. 0x10
   E. 0x15

4. Which of the following programming concepts would be best used for “branching”? (circle all that apply)
   A. 'if' statements
   B. 'while' loop
   C. 'for' loop
   D. methods/functions

5. In programming, what is the purpose for writing a function? (circle all that apply)
   A. providing abstract or modular design to allow for greater code re-use/functionality
   B. provides a way to call code that may need to be run multiple times
   C. providing an algorithm that solves a specific task
   D. breaking the program into smaller sub-problems that are easier to write
Short Answer (10 points):
Instructions: Answer the following with as much detail as possible in one to two sentences.

1. Memory can be broken down into two primary components, the “stack” and the “heap”. Explain what data is stored in each.

Stack stores primitive data types as well as references to arrays and objects. Heap stores the data associated with arrays and objects.

2. What is the difference between a binary search and a linear search?

Binary search is a divide and conquer algorithm that is used to search a sorted list of items. Binary search uses splits a list in half comparing the middle element to see if it’s the item being searched, if not, it eliminates the half of the list that is either larger or smaller than the item being searched for. Its runtime performance is O(log(n)). Linear search can work with unsorted data but it compares every element until a match is found. As such, its runtime performance is O(n) which is substantially slower than binary search.

3. What is an off-by-one error?

A logic error usually found when a boundary condition has been violated. An example of which you iterate through every element in an array and instead of starting at index 0, the code starts at index 1. Or possibly indexing past the last element in the array.

4. Explain what “overloading” is and what type of construct would be overloaded.

Defining multiple functions with the same name, but different parameters. Functions can be overloaded.

5. Explain the difference between an 'index' and an 'element' in an array.

In an array multiple values or elements are stored. Each element is accessed using a corresponding numeric index. The index is the position of that element in the array.

| Index: | 0 | 1 | 2 | ... | N-1 |
---|---|---|---|---|---|
| Element: | obj1 | obj2 | obj3 | ... | objN |
Entomology – Study of Bugs (10 points):

Instructions: Find 3 bugs in the following program (there are more than three):
   1) Specify the line number of the bug.
   2) Check the box next to the type of error
   3) Write a short explanation for why this is an error or a fix for the error.

Note: There can be more than one bug per line. Assume all comments correctly describe what should be occurring in the program.

```
import Scanner;

class BugsEven {
    public static final char QUIT = "0";
    public static void isEven(int num) {
        return (num % 2 != 1);
    }
    public static int main(String[] args) {
        // Get input
        Scanner scanner = Scanner(System.in);
        System.out.println("Enter a number (0 to quit): ");
        int input = scanner.nextInt();
        // Run until QUIT is entered
        while (input != QUIT) {
            if (isEven(input)) {
                System.out.println(input, " is even");
            } else
                System.out.println(input + " is odd");
        }
        // Get next input
        System.out.println("Enter a number (0 to quit): ");
        String input = scanner.nextInt();
        scanner.done();
    }
}
```

Line 1: Syntax: java.util.Scanner
Line 5: Syntax: public static final int QUIT = 0;
Line 8: Syntax: void → boolean
Line 13: Logic: int → void
Line 15: Syntax: new Scanner(System.in)
Line 17: Syntax: scanner.nextInt()
Line 21: Syntax: input + " is even"
Line 27: Syntax: input = scanner.nextInt() // input type already declared
Line 29: Syntax: scanner.close()
What is the Output? (10 points): Variable Scope

Instructions: What is the output that would be printed to the screen with the following code?

Hint: Write down all your different variables (keep in mind the scope of those variables) and update their values as you step through the code.

```java
public class Output {
    static final String WELCOME = "Subway Scope!";
    static char[] subwayLines = { 'B', 'D', 'F' }; 
    static int currentLine = 0;

    static char getline() {
        return subwayLines[currentLine];
    }
    static void printLine(int i) {
        System.out.println(subwayLines[i]);
        i++;
    }
    static char printLines(char[] subwayLines) {
        for (char train: subwayLines) {
            System.out.print(train);
        }
        System.out.println();
    }

    public static void main(String[] args) {
        char[] subwayLines = { 'A', 'C', 'E' }; 
        System.out.println(WELCOME);
        printLines(subwayLines);

        for (currentLine = 0; currentLine < Output.subwayLines.length; currentLine++) {
            System.out.println("line: " + getline());
        }

        int i = 0;
        System.out.println("i: " + i);
        printLine(i);
        System.out.println("i: " + i);
    }
}
```

Answer:
Subway Scope!
ACE
line: B
line: D
line: F
i: 0
B
i: 0
Fill in the Code – Hip Hip Array! (10 points):

Instructions: Fill in the missing code so that the code does what it is described in the comments.
Note: Watch out for off-by-one errors and blanks could be operators (equivalents, arithmetic, boolean)

```java
import java.util.Scanner;

/**
 * Pulls a student at random from a list of students entered (one student per line) from Standard Input.
 *
 * This program is most easily usable by passing a student roster on the CLI
 * using input redirection:
 * <pre>
 * java RandomStudent < INPUTFILE.txt
 * </pre>
 */

public class RandomStudent {

    // Constant number of lines incoming from Standard Input
    public static final int NUM_ENROLLED = 60;

    public static void main(String[] args) {
        // initialize an empty array (must be big enough to store all students)
        String students[] = new String[NUM_ENROLLED];

        // Populate list of students from standard input
        Scanner input = new Scanner(System.in);
        int studentCount = 0;

        // Run until no more lines exist from standard input
        while (input.hasNextLine()) {
            // Get the name of the next student and store it
            String studentName = input.nextLine();
            students[studentCount] = studentName;
            studentCount++;
        }

        // Get a student at random
        int randomStudent = (int) (Math.random() * NUM_ENROLLED);

        System.out.println("The chosen one: " + students[randomStudent]);

        // Good housekeeping (close opened scanners)
        input.close();
    }
}
```
Write the documentation - (10 points):

Fill in appropriate comments for both /** ... */ comment blocks and // comment lines.

Hint: Start in the main() and step through documenting the code as you go.

```java
/**
 * Creates a duplicate array that is identical to the original array, except
 * that a new element has been appended to the beginning of the array. All other
 * elements are moved up one index (value at index 0 moved to index 1). The
 * original array is not modified. The new array will have a length of 1
 * more than the original array.
 * @param array
 *            the original array
 * @param value
 *            the value of the element to add
 * @return a new array with an added element at the start of the array
 */

public static int[] push(int[] array, int value) {
    // Create new array
    int[] newArray = new int[array.length + 1];
    // Assign new starting value
    newArray[0] = value;
    // Copy all remaining values
    for (int i = 1; i < newArray.length; i++) {
        newArray[i] = array[i - 1];
    }
    return newArray;
}

/**
 * Creates a duplicate array that is identical to the original array,
 * except that the first element from the array has been removed. All other
 * elements are moved down one index. The original array is not modified.
 * The new array will have a length of 1 less than the original array.
 * @param array
 *            the original array
 * @return a new array with first element removed.
 */

public static int[] pop(int[] array) {
    // Create new array
    int[] newArray = new int[array.length - 1];
    // Copy all but the first element
    for (int i = 1; i < array.length; i++) {
        newArray[i - 1] = array[i];
    }
    return newArray;
}
```
Write the Code (10 points):
Using either pseudocode or Java code, write code that prints the multiples of 3 starting at 3333 and going to 3 inclusively. No class definition is necessary, just the write code that would be inside the main().

Note: Don’t hardcode values into your program logic and be careful of off by one errors.

```java
start = 3333
end = 3
multiple = 3
for (i = start; i > end; i = i – multiple)
    print(i)
```

```java
static final int START = 3333;
static final int END = 3;
static final int MULTIPLE = 3;
for (int i = START; i > END; i -= MULTIPLE) {
    System.out.println(i);
}
```

```java
static final int START = 3333;
static final int END = 3;
static final int MULTIPLE = 3;
for (int i = START; i > END; i--) {
    if (i % MULTIPLE == 0) {
        System.out.println(i);
    }
}
```
Write the Code (10 points):
Instructions: In Java code, write the function count() that completes this code. Comments are not necessary, but can earn you partial credit.

```java
/**
 * Count the number of occurrences of a value in an array
 * @param array the array to analyze
 * @param valueToFind the value to search for
 * @return the number of occurrences of <code>valueToFind</code> in <code>array</code>
 */
public static int count(int[] array, int valueToFind) {
    int count = 0;
    for (int value : array) {
        if (value == valueToFind) {
            count++;
        }
    }
    return count;
}
or:
    count = 0;
    for (int i = 0; i < array.length; i++) {
        if (array[i] == valueToFind) {
            count++;
        }
    }
    return count;
```
Write the Code (10 points):

Instructions: In Java code, write the function that matches the following specification. Comments are not necessary, but can earn you partial credit. No main() is required, just the function definition.

Hint: Consider writing a helper function to simplify the program

```java
/**
 * Prints numbers in this format when called with printNums(3):
 * <pre>
 * 1
 * 12
 * 123
 * 123
 * 12
 * 1
 * </pre>
 *
 * @param number the number to count up to and print down from.
 */
public static void printNums(int n) {
    // counting up
    for (int row = 1; row <= n; row++) {
        for (int col = 1; col <= row; col++) {
            System.out.print(col);
        }
        System.out.println();
    }
    // counting down
    for (int row = n; row >= 1; row++) {
        for (int col = 1; col <= row; col++) {
            System.out.print(col);
        }
        System.out.println();
    }
}
```