

**Name:**  
**Tutor:**  
**Student ID:**  
**Signature:**

**CSE 8B**  
**Practice Final**  
**Fall 2015**

You're allowed one page (front and back) of hand written notes to use during the final. Bring your student ID. Final is on Monday, 7-10pm in CNTR 105.

1) What is output of the following logic

```
int i = 3;
while (i <= 8) {
    if (i % 2 == 0 && i != 7) {
        i += 3;
    } else {
        i--;
    }
    System.out.println("i = " + i);
}
```

2) Which of the following are valid Java identifiers?

|         |        |              |        |
|---------|--------|--------------|--------|
| \$ocks! | CSE-8B | 777sevens777 | 0xFFFF |
| CSE_8B  | Int    | JavaClass    | new    |

3) What gets printed by this code?

```
public class FinalExam
{
    public static void main(String[] args)
    {
        int[] array = { 35, 85, 55, 115, 155, 95, 35, 155, 45 };
        int x = 73, y = 0;

        for (int i = 0; i < array.length; i++) {
            int n = array[i];

            if (n < x) {
                x = n;
                y = i;
            }
        }

        System.out.println(x + ", " + y);
    }
}
```

What is printed if the line `if (n < x)` was changed to `if (n <= x)`?

What is printed if the line `if (n < x)` was changed to `if (n >= x)`?

What is printed if the line `if (n < x)` was changed to `if (n > x)`?

What is printed if the line `if (n < x)` was changed to `if (n == x)`?

What is printed if the line `if (n < x)` was changed to `if (n != x)`?

#### 4) What gets printed:

```
recurse(5);
public void recurse(int x) {
    if (x<0)
        System.out.println("complete");
    else {
        System.out.println(x);
        recurse(x-1);
    }
}
```

```
recurse(0);
public void recurse(int x) {
    if (x < 5)
        System.out.println("complete");
    else {
        System.out.println(x);
        recurse(x+1);
    }
}
```

```
recurse(5);
public void recurse(int x) {
    if (x<=0)
        System.out.println("complete");
    else {
        System.out.println(x);
        recurse(x-1);
        System.out.println(x);
    }
}
```

```
recurse(5, 2);
public void recurse(int x, int y) {
    if (x <= 0)
        System.out.println("complete");
    else {
        System.out.println(x-y);
        if (y%2 == 0)
            y = 1;
        else
            y = 2;
        recurse(x-1, y);
    }
}
```

```
recurse(5);
public void recurse(int x, int y) {
    if (x > 0)
        System.out.println("complete");
    else {
        System.out.println(x);
        recurse(x-1);
    }
}
```

```
recurse(0);
public void recurse(int x) {
    if (x > 0)
        System.out.println("complete");
    else {
        System.out.println(x);
        recurse(x-1);
    }
}
```

5) Which of the following situations would multithreading be useful?

Displaying text to the terminal in a specific order

Calculating a sequence of values, each of which require the previous result

Adding an array of integers together

Animating multiple objects

6) Given that *MyThread* extends *Thread* how would you start executing the thread :

*Thread t1 = new MyThread();*

7) Circle the properties that are true for the following:

#### **Abstract Classes -**

- A) Can have constructors
- B) Does not always have abstract methods
- C) An instance of an abstract class may be created
- D) Inherited by using *extends* keyword
- E) Fields must be constants (i.e. public, static, final)
- F) A class can inherit multiple abstract classes

#### **Interfaces -**

- A) Can have constructors
- B) All methods must be abstract
- C) An instance of an interface may be created
- D) Inherited by using *implements* keyword
- E) Fields must be constants (i.e. public, static, final)
- F) A class can inherit multiple interfaces

8) Given the following definitions:

```
public interface Animal {  
    public abstract String speak(String str);  
}
```

```
class Cat implements Animal {  
    private String speak;  
  
    public Cat() {  
        this.speak = "meow";  
    }  
  
    public String speak(String str) {  
        return str + ": " + this.speak;  
    }  
  
    public String speak() {  
        return this.speak("Cat");  
    }  
}
```

```
class Dog implements Animal {  
    private String speak;  
  
    public Dog() {  
        this.speak = "woof";  
    }  
  
    public String speak(String str) {  
        return str + ": " + this.speak;  
    }  
  
    public String speak(boolean pb) {  
        if (pb) {  
            return this.speak("Peanut");  
        } else {  
            return this.speak("Butter");  
        }  
    }  
}
```

And the following variable definitions:

```
Cat cat = new Cat();  
Dog dog = new Dog();  
Animal animal;
```

What gets printed with the following statements (each statement is executed in the order it appears). If there is a compile time error, write "Error" and assume that line is commented out when run.

```

System.out.println(cat.speak());
System.out.println(cat.speak(true));
System.out.println(cat.speak("Cocoa"));
System.out.println(dog.speak());
System.out.println(dog.speak(true));
System.out.println(dog.speak("Boba"));
animal = cat;
System.out.println(animal.speak());
System.out.println(animal.speak(true));
System.out.println(animal.speak("Cocoa"));
animal = new Dog2();
System.out.println(animal.speak());
System.out.println(animal.speak(true));
System.out.println(animal.speak("Boba"));

```

## 9) List the Big O complexity

```

for (int i = 0; i < n; i++) {
    for (int j = 0; j < 10000; j++) {
        for (int k = 0; k < n/2; k++) {
            System.out.println(i+j+k);
        }
    }
}

```

```

int index = n/2;
while (index != 0) {
    index = index/2;
}

```

```

for (int i = 0; i < n; i+=2) {
    for (int j = i; j < n/2; j+=2) {
        System.out.println(i+j);
    }
}

```

```

for (int i = n; i > 0; i--) {
    for (int j = Integer.MAX_VALUE; j > i; j--) {
        System.out.println(i+j);
    }
}

```

```

for (int i = n; i > n-100; i--) {
    for (int j = 0; j < (n-n/2); j++) {
        System.out.println(i+j);
    }
}

```

```

for (int i = n; i < (n+100); i++) {
    for (int j = n; j > (n-100); j--) {
        System.out.println(i+j);
    }
}

```

10) Given the following path:

`/home/jundt/cse8b/grades/final/cs8bxx.txt`

And that the home directory is `/home/jundt`, answer the following questions.

What command shows the current working directory?

Assume the command returns `/home/jundt/cse8b/grades`. How do you open `cs8bxx.txt` without changing the directory? Use the text editor of your choice.

You change your final grade in the text file to 100%. How do you save and close the text editor that you opened the file with? For vim users, assume that you are still in “insert” mode.

Still in `/home/jundt/cse8b/grades`, you type a thank you note to Professor Jundt. How do you verify the text file was successfully created? More specifically, what command do you use to show the contents of the current directory?

To be sure Professor Jundt sees the file, you decide to put it in the `/home/jundt/cse8b` directory. How do you change your current working directory to `/home/jundt/cse8b`?

How do you copy the thank you note from `/home/jundt/cse8b/grades` to the current directory, `/home/jundt/cse8b`? The note is named *thankyou.txt*.

Upon second thought, you decide to remove the original *thankyou.txt* from `/home/jundt/cse8b/grades`. What command would you use to accomplish this?

You realize that `/home/jundt` would be a better directory to put the thank you note in. How do you move *thankyou.txt* there? Remember that you are currently in `/home/jundt/cse8b`.

11) Write what gets printed.

```
int[] array = {0, 1, 2};
doStuff(array);
System.out.println(array[0]);
System.out.println(array[1]);
System.out.println(array[2]);

public void doStuff(int[] arr) {
    arr[0] = arr.length;
    int x = arr[2];
    x++;
    arr[1] = x;
}
```

12) List in order the methods that run.

```
try {
    method1();
    method2();    // causes an ArrayIndexOutOfBoundsException
    method3();
}
catch (Exception e) {
    method4();
    method5();
}
method6();
```

```
try {
    method1();    // causes a NullPointerException
    method2();
}
catch (Exception e) {
    method3();
    method4();
}
finally {
    method5();
}
method6();
```

```
try {
    method1();
    method2();
    method3();    // causes an ArithmeticException
}
catch (Exception e) {
    method4();
    return;
}
finally {
    method5();
}
method6();
```

13) List the best data structure for the given situation.

array      ArrayList      HashSet      HashMap

Storing a phone number along with a name.

Storing an unknown number of things in order

Storing words in a dictionary, so you can efficiently check them later on.

Storing items that you will have to iterate through.

Storing something without importing anything.

Storing exactly 100 numbers, so you can sort them later

14) Sort the following based on where the content of the variable is stored. Remember the difference between a pointer that references that content versus the actual content itself.

String      int      char      int[]      Scanner      float      String[]

15) Which method would be used here in order to pass these tests? (Assume the latest version of JUnit is in the build path).

```
@Test
public void test1() {
    Integer i1 = 10;
    Integer i2 = 90;

    _____(i1 + i2, new Integer(100));
}

@Test
public void test2() {
    String s1 = "CSE8B";
    String s2 = new String("CSE8B");

    _____(s1, s2);
}
```

In test-driven development, unit-tests should be written after the code. (TRUE / FALSE)

After all of the unit tests pass, the program is fully functional and ready to ship. (TRUE / FALSE)

Unit tests are designed to test a single entity of the program (a class or method). (TRUE / FALSE)

16) Draw the UML for the following:

```
public class FinalExam {
    private String name;
    private int grade;
    private boolean crying;
    public FinalProject() {...}
    public static boolean setGrade(int grade) {...}
    public void beginCrying() {...}
}

public abstract class Student {
    public static String university;
    public abstract String getName() {...}
}

public class You extends Student {
    public int age;
    public You() {...}
    public boolean setAge(int age) {...}
}
```

17) What gets printed by this code?

```
class ClassA {
    public void method1() {
        method3();
        System.out.println("ClassA 1");
    }

    public void method3() {
        System.out.println("ClassA 3");
    }
}

class ClassB extends ClassA {
    public void method1() {
        super.method3();
        System.out.println("ClassB 1");
    }

    public void method2() {
        method1();
        System.out.println("ClassB 2");
    }
}

class ClassC extends ClassA {
    public void method1() {
        super.method3();
        System.out.println("ClassC 1");
    }

    public void method3() {
        System.out.println("ClassC 3");
    }
}

class ClassD extends ClassB {
    public void method2() {
        super.method2();
        System.out.println("ClassD 2");
    }

    public void method3() {
        super.method1();
        System.out.println("ClassD 3");
    }
}
```

```
public class Test1 {
    public static void main(String[] args) {
        ClassA ref1;

        ref1 = new ClassD();
        ( (ClassB) ref1 ).method1();
        System.out.println("-----");
        ref1.method3();
        System.out.println("-----");
        ( (ClassD) ref1 ).method2();
    }
}
```

```
public class Test2 {
    public static void main(String[] args) {
        ClassA ref1;

        ref1 = new ClassC();
        ref1.method1();
        System.out.println("-----");
        ( (ClassC) ref1 ).method2();
        System.out.println("-----");
        ref1.method3();
    }
}
```

18) Given the following Java classes, select which would be best to use in the following situations:

|                 |                     |                      |
|-----------------|---------------------|----------------------|
| System (in/out) | BufferedInputStream | BufferedOutputStream |
|                 | Scanner             | PrintWriter          |

- |                                   |       |
|-----------------------------------|-------|
| 1. Reading text from a file       | _____ |
| 2. Writing to a file byte-by-byte | _____ |
| 3. Reading to a file byte-to-byte | _____ |
| 4. Writing text to the terminal   | _____ |

19) Fill in each blank with one of the following key words: *Exception, compiler error, package, GUI, IDE, UML, constant, interface, abstract class, polymorphism, text editor*

- If an error occurs before runtime, it is considered a(n) \_\_\_\_\_, but if it occurs during runtime, it is a(n) \_\_\_\_\_.
- An interface that is more than just plain text (buttons, canvas, text boxes, etc.) is a(n) \_\_\_\_\_.
- VI and emacs are examples of a dedicated \_\_\_\_\_.
- Eclipse and IntelliJ are examples of a(n) \_\_\_\_\_.
- A(n) \_\_\_\_\_ is usually declared as both static and final.
- The *implements* keyword is associated with \_\_\_\_\_(s), while the *extends* keyword is associated with \_\_\_\_\_(s).