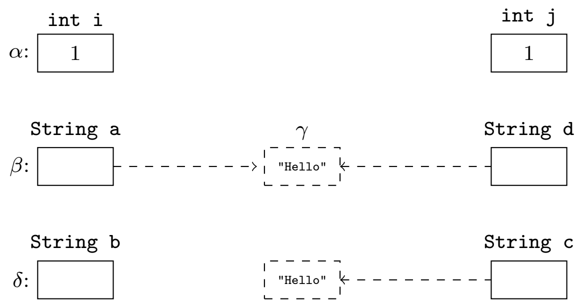
Final

CS3120-MW2

May 25th, 2016

Total Points: 120

1 **(6 points) Given the following image, write a set of declarations that correspond to it.** Do not write a `main()` method, simply write the declarations. **Note:** there is more than one right answer. (Greek letters are only for question (2)).



(answer below this line)

```
int i = 1;
int j = 1;
String a = "Hello";
String b;
String c = new String("Hello");
String d = a;
```

2 **(10 points) Given the image from (1), choose one of the following options for each question.**

C	α **is a(n):** (a) object (b) reference (c) primitive
B	α **is dynamically allocated:** (a) true (b) false
B	β **is a(n):** (a) object (b) reference (c) primitive
B	β **is dynamically allocated:** (a) true (b) false
C	β **holds a(n):** (a) object (b) boolean (c) address
A	γ **is a(n):** (a) object (b) reference (c) primitive
A	γ **is dynamically allocated:** (a) true (b) false
B	δ **is a(n):** (a) object (b) reference (c) primitive
B	δ **is dynamically allocated:** (a) true (b) false
C	δ **may hold a(n):** (a) object (b) boolean (c) address

3 **(8 points) Given the image from (1) and the following expressions, determine if they are true or false.** If the expression is invalid (i.e. compile error, runtime error, etc.) write: "error".

`i == j`	?	true
`a == d`	?	true
`c == d`	?	false
`b == c`	?	false
`i.equals(j)`	?	error
`a.equals(d)`	?	true
`d.equals(c)`	?	true
`b.equals(c)`	?	error

4 **(2 points) Consider the following code:**

```
Object o = new Object();
Object p = null;
Object q = o;
o = p;
```

An `Object` was allocated above. Is it elegible for garbage collection? If not, write some code below to make it eligible for garbage collection.

No, the `Object` is still referenced by `q`. To remove all references: `q = null`.

5 **(2 points) Consider the following code.**

```
void swapFirst(String [] first, String [] second) {
    String tmp = first[0];
    first[0] = second[0];
    second[0] = tmp;
}
```

```
String [] s = new String [] { "Hello" }
String [] r = new String [] { "Hi" }
swapFirst(s, r);
System.out.println(s[0] + " " + r[0]);
```

What is printed by the last line above?

```
Hi Hello
```

6 **(2 points) Consider the following code.**

```
void swap(String first, String second) {
    String tmp = first;
    first = second;
    second = tmp;
}
```

```
String s = "Hello";
String r = "Hi";
swap(s, r);
System.out.println(s + " " + r);
```

What is printed by the last line above?

```
Hello Hi
```

7 **(2 points) What is the difference between a mutable and an immutable object?**

An immutable object is one whose state – i.e. variables – cannot be changed after creation. This means, neither the references nor the objects to which the references point can change.

A mutable object, on the other hand, may change after creation.

8 **(4 points) What is the value of the `String` after each line?**

`String s = "Hello";`	s = `Hello`
`s.substring(2);`	s = `Hello`
`s.toLowerCase();`	s = `Hello`
`s = s.concat("?");`	s = `Hello?`

9 **(1 point) Consider the following class.**

```
class Foo {
    int get() { return i; }
    private int i = 1;
}
```

Is this class mutable or immutable? If it is mutable, change the `get()` method so that it is immutable.

This class is immutable as the instance variable `i` is private and cannot be accessed. Furthermore, the `get()` method returns a *copy* of the value and *not a reference*.

10 **(3 points) Consider the following class.**

```
class Foo {
    List<Integer> get() { return l; }
    private final List<Integer> l = new Vector<Integer>();
}
```

Is this class mutable or immutable? If it is mutable, change the `get()` method so that it is immutable.

This class is mutable as the elements in the `List` can be changed through the reference returned in the `get()` method. The revised method below makes a copy of the `List` and returns a reference to that copy.

```
List<Integer> get() {
    List<Integer> temp = new Vector<Integer>();
    temp.addAll(l);
    return temp;
}
```

11 **(6 points) Given the following code which specific types of `Exception` might be thrown?** Write whether each is *checked* or *unchecked*. Hint: the method signature for `parseInt()` is:

```
public static void parseInt(String s) throws NumberFormatException
```

```
class Divide {
    public static void main(String [] args) {
        int dividend = Integer.parseInt(args[0]);
        int divisor  = Integer.parseInt(args[1]);
        int quotient = dividend / divisor;
        System.out.println(quotient);
    }
}
```

(answer below this line)

`NumberFormatException` – *unchecked*
`ArrayIndexOutOfBoundsException` – *unchecked*
`ArithmeticException` – *unchecked*

12 **(4 points) Write whether each of the following is *checked* or *unchecked*.**

`Exception`	?	checked
`Error`	?	unchecked
`RuntimeException`	?	unchecked
`Throwable`	?	checked

13 **(9 points) Determine whether the following statements are true or false.**

true	All classes inherit from `Object`
true	Unbuffered streams read one character at a time
true	All variables in an interface are implicitly `final`
true	`default` methods in an interface must be implemented
true	All variables in an interface are implicitly `static`
false	Buffered streams read one character at a time
true	`static` methods in an interface must be implemented
false	All interfaces inherit from `Object`
true	All methods and variables in an interface are implicitly `public`

14 **(5 points) In the space below, write a class called `Snap` that inherits from `Crackle` and implements `Pop`.** Write it so that no class can inherit from `Snap`. Any methods that need to be implemented can simply contain a `return` statement.

```
interface Pop {
    void a();
    default void b() { return; }
    static void c() { return; }
    void d();
}

abstract class Crackle {
    public void a() { return; }
}
```

(answer below this line)

```
final class Snap inherits Crackle implements Pop {
    public void d() { return; }
}
```

15 **(6 points) Given the code you've written above, determine which object can be assigned to the given reference variable. Then, write whether the assignment performs an *implicit* cast, an *explicit* cast, or neither.**

`Snap snap = new`	`Snap()`	neither
`Crackle crackle = new`	`Snap()`	implicit
`Pop pop = new`	`Snap()`	implicit

16 **(4 points) Determine whether each of the following is an *upcast*, a *downcast*, or neither.**

`String s = "Hello";`	neither
`Object o = s;`	upcast
`String r = (String)o;`	downcast
`Object p = new Vector();`	upcast

17 **(6 points) Write a class called `Foo` that is a subclass of `Bar`.** `Foo` should have an instance variable `x` which is different than that inherited from `Bar`. Implement a method called `incrementBar()` which increments the value of the instance variable `x` inherited from `Bar`. Also, implement a proper constructor for `Foo` that makes use of `Bar`'s constructor.

```
class Bar {
    Bar(int x) { this.x = x; }
    int x;
}
```

(answer below this line)

```
class Foo extends Bar {
    Foo(int x) { super(x); }
    void incrementBar() { super.x++; }
    int x;
}
```

18 **(2 points) Consider the code below.** Then, answer the proceeding questions.

```
import java.applet.*;
import java.awt.*;

public class HelloWorld extends Applet {
    public void start() {
        Graphics g = this.getGraphics();
        g.drawString("Hello, World.", 20, 20);
    }
}
```

What will be displayed in the browser when the applet is first rendered?

The text: "Hello, World."

What will be displayed in the browser after the window is minimized and then maximized?

Nothing, the text will have disappeared.

19 **(3 points) Consider the code below.** Then, answer the proceeding questions.

```
import java.awt.*;
import java.applet.*;

public class HiFriend extends Applet {
    String str1 = "Hi, friend.";
    String str2 = "Hi, there.";
    boolean b = true;

    public void paint(Graphics g) {
        if(b) g.drawString(str1, 20, 20);
        else  g.drawString(str2, 20, 20);
        b = !b;
    }
}
```

What will be displayed in the browser when the applet is first rendered?

The text: "Hi, friend."

What will be displayed in the browser after the window is minimized and then maximized?

The text: "Hi, there."

What will be displayed if we minimize and maximize another time?

The text: "Hi, friend."

20 **(4 points) What is the difference between low-level events and high-level (or, semantic) events?**

Low-level events generally refer to individual input and output events related to underlying hardware – i.e. mouse clicks, key presses, etc.

High-level events generally refer to some event within the GUI which is not necessarily triggered by a single or specific low-level event – i.e. a button on a GUI can be pressed via a mouse click, clicking enter on the keyboard, tapping a touch screen, etc. However, in each scenario it simply makes sense to speak of a button click regardless of the triggering low-level event.

21 **(2 points) What two peices of information are required to open a socket?**

An IP address and a port number.

22 **(2 points) What are the two ways to create threads using the Java libraries?**

(1) `extend` the `Thread` class.
(2) `implement` the `Runnable` interface.

23 **(1 point) What is a context switch?**

When the processor switches from one thread (or, process) to another.

24 **(4 points) Consider the following code.** Then, answer the proceeding questions.

```
class FooBar {
    public static void main(String [] args) throws InterruptedException {
        Thread t = new Thread() {
            public void run() {
                try { Thread.sleep(1); } catch(Exception e) { return; }
                System.out.println("Foo!");
            }
        };

        t.start();
        Thread.sleep(1);
        System.out.println("Bar!");
    }
}
```

What will the code above print?

Foo!
Bar!

Is it guaranteed to print this? Why or why not?

No, this is not guaranteed as `Thread.sleep()` does not guarantee that the thread will remain inactive for the given amount of time.

25 **(5 points) Consider the following code.** Then, answer the proceeding questions.

```
class FooBar {
    public static void main(String [] args) throws InterruptedException {
        Thread t = new Thread() {
            public void run() {
                try { Thread.sleep(1); } catch(Exception e) { return; }
                System.out.println("Foo!");
            }
        };

        System.out.println("FooBar!");
        t.start();
        t.join();
        System.out.println("Bar!");
    }
}
```

What will the code above print?

FooBar!
Foo!
Bar!

Is it guaranteed to print this? Why or why not?

Yes, this is guaranteed as `t.join()` guarantees that `main()` thread will not execute until thread `t` returns.

26 **(4 points) Which of the following code segments executes with a race condition?** Assume the following variables have been declared.

```
int i;
final boolean b = false;
```

`i = 10;`	||	false
`i--;`	||	true
`if(b) x = 1;`	||	false (`b` *always* false)
`if(!b) x = 2;`	||	true

27 **(3 points) Consider the code below.** Then, answer the proceeding questions.

```
class Blah implements Runnable {
    static int i = 0;
    public void add() { i++; }
    public void run() { add(); }
}

class Demo {
    public static void main(String [] args) {
        Thread t = null;
        for(int x = 0; x < 10; x++) {
            t = new Thread(new Blah());
            t.start();
        }
    }
}
```

Why is the code above problematic in a multithreaded environment? Explain.

The code above contains a race condition. Namely, when it executes the following statement in the `add()` method: `i++`. Since this statement contains more than one operation – i.e. is not atomic – a context switch is liable to occur while the statement is executing. If the context switch executes some other thread which makes a call to the same method, the data is liable to be corrupted.

How can you fix the problem? Explain.

Use the `synchronized` modifier to ensure only a single thread is executing the method at a time.

28 **(10 points) Determine whether the following statements are true or false.**

false	`MouseEvent`s are high-level events
false	GUI components send events directly to their listeners
true	Immutable objects are thread safe
true	`ItemEvent`s are high-level events
false	A client is said to listen on a port
true	All GUI events in Java are handled with a single thread
true	`KeyEvent`s are low-level events
true	A server is said to listen on a port
false	Mutable objects are thread safe
false	`ActionEvent`s are low-level events