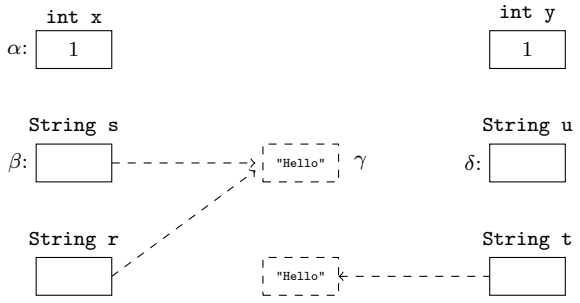
Midterm

CS3120-MW2

April 6th, 2016

Total Points: 113

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 x = 1;
int y = 1;
String s = "Hello";
String r = s;
String t = new String("Hello");
String u;
```

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".

`x == y`	?	true
`s == r`	?	true
`r == t`	?	false
`u == t`	?	false
`x.equals(y)`	?	error
`s.equals(r)`	?	true
`r.equals(t)`	?	true
`u.equals(t)`	?	error

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

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

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

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

5 **(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
```

6 **(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
```

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

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

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

```
class Foo {
    String get() { return s; }
    private final String s = "Hello";
}
```

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

This class is immutable, as `String`s are immutable in Java. This means it is safe to `return` a reference to the `String` object in the `get()` method.

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

```
class Foo {
    int [] get() { return arr; }
    private final int [] arr = { 1, 2, 3 };
}
```

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

This class is not immutable as the elements in the array can be changed through the reference returned in the `get()` method. The revised method below makes a copy of the array and returns a reference to that copy.

```
int [] get() {
    int [] x = new int[arr.length];
    for(int i = 0; i < arr.length; i++)
        x[i] = arr[i];
    return x;
}
```

11 **(7 points) Given the following code, what are the values of the proceeding variables?** If they are uninitialized, write: "uninitialized".

```
class Foo {
    static int a;
    boolean b;
    int [] c;

    void f() {
        int d;
        boolean e;
        int [] f;
    }

    public static void main(String [] args) {
        String g;
    }
}
```

a = 0

b = false

c = null

d = uninitialized

e = uninitialized

f = uninitialized

g = uninitialized

12 **(2 points) What is the difference between a default constructor and a no-argument constructor?**

A **default constructor** is the constructor provided by the Java compiler which has a single line:

```
super();
```

Coincidentally, it takes no arguments.

A **no-argument constructor** is one which does not take any arguments.

13 **(2 points) What are two benefits of static factory methods?**

(1) Static factory methods provide a means of obtaining a reference to an object of a particular class before any objects of that class exist. While this can be done using constructors, static factory methods may have descriptive names allowing for two factory methods to have identical parameter lists. Constructors are unnamed and therefore, two constructors in the same class cannot have the same parameter list.

(2) Static factory methods allow input validation before creating an object so that the constructor does not throw an exception and allocate memory for an unusable object.

14 **(6 points) Fill in the access chart below.** Write 'Y' if the modifier allows access in that particular context or 'N' if it does not allow access – i.e. `public` allows access everywhere, so all columns recieve a 'Y'.

Modifier	Class	Package	Subclass	World
`private`	Y	N	N	N
no modifier	Y	Y	N	N
`protected`	Y	Y	Y	N
`public`	Y	Y	Y	Y

15 **(1 point) Given the following code, what is the fully qualified name of the class `HelloWorld`.** (This means the *full* name of the class, incl. package information).

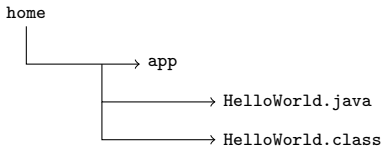
```
package app;

class HelloWorld {
    public static void main(String [] args) {
        System.out.println("Hello, World");
    }
}
```

(answer below this line)

`app.HelloWorld`

16 **(1 point) Given the class above, and the directory structure below, write the command you would execute from the command line in order to run `HelloWorld`.** Assume you are in the directory called `home`.



(answer below this line)

```
java app.HelloWorld
```

17 **(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*

18 **(4 points) Write whether each of the following is *checked* or *unchecked*.**

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

19 **(2 points) What are the two most generalized character stream classes?**

`Reader` and `Writer`

20 **(2 points) What are the two most generalized byte stream classes?**

`InputStream` and `OutputStream`

21 **(9 points) Determine whether the following statements are true or false.**

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

22 **(3 points) In the space below, write a class called `Scooby` that inherits from `Dooby` and implements `Doo`.** Write it so that no class can inherit from `Scooby`.

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

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

(answer below this line)

```
final class Scooby inherits Dooby implements Doo { }
```

23 **(3 points) Given the code you've written above, fill in the blanks below:**

```
Scooby scooby = new ____Scooby();____
  Dooby dooby = new ____Scooby();____
      Doo doo = new ____Scooby();____
```

24 **(3 points) Write a class called `Foo` that is a subclass of `Bar`.** All you need to implement is 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); }
}
```

25 **(1 point) Consider the following class:**

```
class SomeClass {
    void f() throws IOException { throw new IOException(); }

    public static void main(String [] args) throws Exception {
        try{
            throw new RuntimeException();
            SomeClass sc = new SomeClass();
            sc.f();
        } catch(IOException ioe) {
            throw ioe;
        }
    }
}
```

What kind of `Exception` is thrown out of `main()`?

`RuntimeException`

26 **(3 points) Write an interface called `Shape` that has a single, unimplemented method called `area()` that takes no parameters and returns a `double` representing the area of the shape. Add a single variable called `PI` with the value `3.1415`.**

```
interface Shape {
    double area();
    double PI = 3.1415;
}
```

27 **(19 points) Write a class called `Circle` that implements your `Shape` interface above and additionally has:**

- a single `private` field of type `double` called `radius`.
- Implement a single static factory method: `withRadius()` that takes a single argument which is a `double` and returns a `Circle` object. If the argument is negative, the method should throw an `Exception`. *Be sure that the only way a `Circle` object can be created is through the static factory methods!*
- Package your class in a package called `shapes`.
- Make only the class and the static factory methods accessible outside the package.

Hint: *remember that the area of a circle is* πr^2 *– this information is relevent.*

```
package shapes;

public class Circle implements Shape {
    private Circle(double r) { radius = r; }

    public static Circle withRadius(double r) throws Exception {
        if(r < 0) throw new Exception("Radius must be non-negative.");
        Circle c = new Circle(r);
        return c;
    }

    public double area() { return radius*radius*PI; }

    private double radius;
}
```