CSE-1020 Midterm

Sur / Last Name:
Given / First Name:
Student ID:
Section: A E

- **Instructors:** Parke Godfrey & Gordon Turpin
- **Exam Duration:** 120 minutes
- **Term:** Fall 2008

1. The exam has six sections with five multiple-choice questions each. The examination time is two hours.
2. Choose *one* best answer for each question.
   - You will receive no point for an unclear choice, no choice made, or multiple answers selected.
   - There is no additional deduction for a wrong answer.
   - Do not use red ink.
3. No aids (such as calculator, reference sheets, textbooks, etc.) are permitted.
4. Please turn off cell phones, and put your cell phones off the desk.
5. Generally, no questions regarding the interpretation, intention, or further elucidation of an exam question will be answered by invigilators. If truly in doubt, state your interpretation next to your answer.
6. Assume that any program fragment is properly within a `main` method within a `class`, and that any unseen part of the program is correct. Thus, it would compile and run, except perhaps for the program fragment.
   Choose **error** if the program would fail either at compile time or at run time.

Grading Box	
1.	/ 5
2.	/ 5
3.	/ 5
4.	/ 5
5.	/ 5
6.	/ 5
Total	/30

1. **Types & Operators.** *What type of operator are you?*

(a)
```
int x = Integer.MAX_VALUE;
System.out.println(x + 1 >= Integer.MAX_VALUE);
```

A. `true`
B. `false`
C. `0`
D. Not enough information to determine.
E. **error**

(b)
```
int x = 2;
int y = 3;
int z = 4;
if ((x * y) / z == x * (y / z))
{
    System.out.println("same");
} else
{
    System.out.println("different");
}
```

A. `same`
B. `different`
C. `same`
`different`
D. Not enough information to determine.
E. **error**

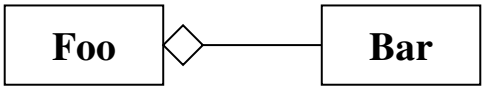
(c)
```
int i = 5;
int j = i  / 2;
System.out.println(j.toString());
```

A. `2`
B. `"2"`
C. `2.5`
D. `3`
E. **error**

(d)
```
System.out.println((new Integer(3)).toString()
                   + (new Double(2.5)).toString());
```

A. `5`
B. `5.5`
C. `32.5`
D. Not enough information to determine.
E. **error**

(e) Consider the following UML diagram.



A. An object of class Foo contains references to objects of class Bar.
B. An object of class Bar contains references to objects of class Foo.
C. Class Foo is derived from class Bar.
D. Class Bar is derived from class Foo.
E. Class Foo is forbidden to call class Bar.

Accept any answer for this one! This is from Chapter 7, which was off limits for this exam.

2. **Delegation.** *The delegation has now arrived.*

(a) `System.out.println((long)3.55);`

A. `3`
B. `3.5`
C. `3.6`
D. `4`
E. error

(b) `System.out.printf("%06.1f%n", 4.5678);`

A. `0000004.5`
B. `0000004.6`
C. `0004.5`
D. `0004.6`
E. error

Say a class called `Thing` has an overloaded method called `doSomething`. There are three versions of `doSomething`.

I. `void doSomething(int a, double b)`
II. `void doSomething(double a, int b)`
III. `void doSomething(double a, double b)`

For the following method calls, which version of `doSomething` will be be invoked?

(c) `doSomething(3, 3.5);`

A. **I**
B. **II**
C. **III**
D. The compiler will arbitrarily choose.
E. **error**

(d) `doSomething(10., 10.);`

A. **I**
B. **II**
C. **III**
D. The compiler will arbitrarily choose.
E. **error**

(e) `doSomething(10, 10);`

A. **I**
B. **II**
C. **III**
D. The compiler will arbitrarily choose.
E. **error**

3. **APIs.** *Iffen you be 'appy?*

Consider a `Rectangle` class with the following API.

```
public class Rectangle extends java.lang.Object
{
    public  static int colour;
    public  int width;
    public  int height;
    public  Rectangle();
    public  Rectangle(int, int); // sets the width and height
    public  int getArea();
}
```

Assume that none of the methods, if called correctly, fails.

(a)
```
Rectangle r = new Rectangle(7, 42);
System.out.println(r.toString());
```

A. `7x42`
B. `294`
C. Prints whatever the colour of the rectangle is.
D. Prints some representation of the rectangle, but we cannot tell from this what.
E. error

(b)
```
Rectangle r = new Rectangle(7, 42);
System.out.println(r.getArea() / r.width);
```

A. `0`
B. `7`
C. `42`
D. Not enough information to determine.
E. error

(c)
```
Rectangle r = new Rectangle(7, 42);
Rectangle s = new Rectangle(3, 8);
r.colour = 3;
s.colour = 4;
System.out.println(r.colour);
```

A. `0`
B. `3`
C. `4`
D. `7`
E. `294`
F. error

(d)
```
Rectangle r = new Rectangle(7, 42);
Rectangle s = new Rectangle(3, 8);
s = r;
r = null;
System.out.println(s.width);
```

A. `0`
B. `3`
C. `7`
D. Not enough information to determine.
E. **error**

(e)
```
Rectangle r = new Rectangle(7, 42);
{
    r.height += 1;
}
System.out.println(r.height);
```

A. `0`
B. `42`
C. `43`
D. Not enough information to determine.
E. **error**

4. **Objects.** *I object, Your Honour!*

   Consider the class `type.lib.Fraction` in the questions below, as we have used in class.

(a)
```
Fraction f1 = new Fraction(1, 2);
Fraction f2 = new Fraction(3, 6);
System.out.println(f1.equals(f2));
```

A. `0`
B. `true`
C. `false`
D. Not enough information to determine.
E. **error**

(b)
```
Fraction f1 = new Fraction(1, 2);
Fraction f2 = new Fraction(1, 2);
System.out.print(f1 == f2);
```

A. `0`
B. `true`
C. `false`
D. Not enough information to determine.
E. **error**

(c)
```
Fraction f1 = new Fraction(3, 5);
f1 = null;
System.out.println(f1.toString() == null);
```

A. `0`
B. `true`
C. `false`
D. Not enough information to determine.
E. **error**

Consider the following legal java program, which compiles and runs without errors.

```
1  import type.lib.Fraction;
2  public class GarbageCollection
3  {
4      public static void main(String[] args)
5      {
6          Fraction f1 = new Fraction(3, 5);
7          Fraction f2 = new Fraction(1, 2);
8          Fraction f3 = new Fraction(3, 4);
9          System.out.println(f1);
10         f1 = f3;
11         System.out.println(f1);
12         f3 = null;
13         ...
14     }
15 }
```

In "line" 13, any number of statements occur. Call the `Fraction` object created at line 6 object A, the `Fraction` object created at line 7 object B, and the `Fraction` object created at line 8 object C.

(d) Object A could be reclaimed by the garbage collector safely after which line?

A. 6
[B.] 10
C. 11
D. 12
E. 14

(e) Object C could be reclaimed by the garbage collector safely after which line?

A. 8
B. 10
C. 11
D. 12
[E.] 14

5. **Control Structures.** *It's simply a matter of control.*

(a)
```
int n = 0;
for (int i = 0; i < 5; i++)
{
    for (int j = 0; j < i; j++)
        n += i;
}
System.out.println(n);
```

A. 9
B. 10
C. 30
D. 36
E. **error**

(b)
```
int x = 5;
int y = 10;
int z = 15;
if (x < y < z) || (z < y < x)
{
    System.out.println("y middle");
} else if (y < x < z) || (y < z < x)
{
    System.out.println("y smallest");
} else
{
    System.out.println(" y largest");
}
```

A. `y smallest`
B. `y middle`
C. `y largest`
D. `10 middle`
E. **error**

(c)
```
int x;
int y = 5;
if (y >= 5)
{
    x = 2 * y;
}
System.out.println(x);
```

A. `5`
B. `10`
C. `x`
D. `null`
E. **error**

(d)
```
int x = 5;
int y = 5;
int z = 7;
if (x < z && x < y)
{
    if (x <= y && z <= y)
    {
        x *= z;
        x += 1;
    }
} else
{
    if (x * y / z < x * z / y)
    {
        x *= y;
        x += 2;
    } else
    {
        x += 2;
        x *= y;
    }
}
System.out.println(x);
```

A. 27
B. 35
C. 36
D. 37
E. error

(e)
```
int j = 1;
for (int i = 1000; i > 0; i /= 2) j++;
System.out.println(j);
```

A. 11
B. 12
C. 13
D. 1000
E. error

6. **Strings.** *I prefer yarn.*

(a)
```
System.out.println((3 + 4 + "5") == (5 + 2 + "5"));
```

[A.] `true`
B. `false`
C. 75
D. 525
E. error

(b)
```
String a = "89";
System.out.println(Integer.parseInt(a.substring(1, 2)));
```

A. 1
B. 8
[C.] 9
D. 89
E. error

(c)
```
String s = "^ab34";
System.out.println(s.replaceAll("[^0-9]", "X"));
```

A. `XabXX`
[B.] `XXX34`
C. `^abXX`
D. `^XXXX`
E. error

(d)
```
String s1 = "HO";
String s2 = s1;
s1 += s2;
s2 += s1;
System.out.println(s2);
```

A. HO
B. HOHO
[C.] HOHOHO
D. HOHOHOHO
E. HOHOHOHOHO

(e)
```
String s = "aBaaCaaaDaaaaE";
System.out.println(s.replaceAll("a+a+", "X"));
```

A. XBXCXDXE
B. aBaaCaaaDaaaaE
C. aBaaCXDXE
[D.] aBXCXDXE
E. error