CSE-1020 Midterm Exam

Written Portion

Family Name: _____
Given Name: _____
Student#: _____
CSE Account: _____
Section: **A** **E**

Instructors: Parke Godfrey (§A) & Burton Ma (§E)
Exam Duration: 90 minutes
Term: Fall 2009

Instructions

1. Write your answers clearly and succinctly.
   (a) You will receive no point for an unclear anwer.
   (b) There is no additional deduction for a wrong answer.
   (c) Do not use red ink.
2. No aids (such as calculator, reference sheets, textbooks, etc.) are permitted.
3. Please turn off cell phones, and put your cell phones off the desk.
4. 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.
5. For any of the program code, assume that
   (a) any program fragment is properly within a `main` method within a `class`,
   (b) any needed classes have been imported,
   (c) any constructor call has a legitimate matching constructor, and
   (d) that any unseen part of the program is correct.
   Thus, it would compile and run, *except perhaps* because of the program fragment shown.

Grading Box	
1.	/10
2.	/10
3.	/10
4.	/10
5.	/10
Total	/50

1. **General Concepts, Operators, & Types.** [10pt]

   For each of the following,

   - state what the program will print,
   - say that the program will not compile *and* state clearly *what* the problem is, *or*
   - say that the program crashes at runtime *and* explain clearly *why*.

   Note `Math.PI` is a `double` with the value `3.141592653589793`.

   Each is worth two points.

(a)
```
final double HALF_PI = (1 / 2) * Math.PI;
System.out.printf("%.1f%n", HALF_PI);
```

0.0

(b)
```
final double HALF_PI = Math.PI;
HALF_PI /= 2;
System.out.printf("%.1f%n", HALF_PI);
```

program will not compile; cannot re-assign `HALF_PI`

(c)
```
System.out.printf("%.1f%n", Math.PI / 2);
```

1.6

(d)
```
int x;
System.out.printf("%.1f%n", x * Math.PI);
```

program will not compile; `x` has not been initialized

(e)
```
long x = Math.PI * 100;
System.out.println(x);
```

program will not compile; cannot store the `double` value `Math.PI * 100` in a `long` variable

2. **Delegation.** [10pt]

Consider a (fictional) utility class named `FractionUtility`. `FractionUtility` has no attributes (public or private) but it does have several public methods. Consider the contract for a method named `divide`. The method has a parameter of type `type.lib.Fraction`.

```
public static void divide(Fraction f, long c)
```

Divides the `Fraction f` by the value of `c`.

Parameters:
`f` - the fraction to divide
`c` - the value to divide by

Precondition:
`c != 0`

Postcondition:
The denominator of `f` is `c` times larger than it was when the method was invoked.

(a) [2pt] What kind of delegation, if any, does this method provide?

procedural delegation

(b) [3pt] Consider the following snippet of client code:

```
Fraction g = new Fraction(12, 13);
long c = 4;
FractionUtility.divide(g, c);
```

What are the values of `c`, `g.getNumerator()`, and `g.getDenominator()` after the code is run?

4, 3, and 13

(c) `[1pt]` Explain what the term *precondition* means in a method contract.

A proposition that the client must ensure is true prior to invoking a method.

(d) `[1pt]` Explain what the term *postcondition* means in a method contract.

A proposition that the implementer promises will be true on return from a method.

(e) `[2pt]` Consider the following snippet of client code:

```
Fraction h = new Fraction(1, 2);
long c = 0;
FractionUtility.divide(h, c);
```

The client code compiles and runs but causes the state of `h` to become `0/0`. Who is at fault, the client or the implementer, of `divide`? Why?

The client is at fault because the precondition has not been met.

(f) `[1pt]` Analyze the postcondition of the method carefully. Explain why you think this postcondition is sensible or not.

The postcondition is not sensible because the Fraction class maintains fractions in simplest terms. If c divides evenly into the numerator the postcondition will not be met.

3. **API.** [10pt]

Consider the (fictional) class `BetterFraction` that is identical to `type.lib.Fraction` except that it has some extra methods; the summary for these methods is shown below:

1	`void`	`multiply(long c)` Multiplies the fraction by the `long` value `c`.
2	`void`	`multiply(double c)` Multiplies the fraction by the `double` value `c`.
3	`void`	`multiply(BetterFraction f)` Multiplies the fraction by another fraction `f`.
4	`double`	`doubleValue()` Returns the `double` value that is closest to the fraction's numerator divided by its denominator.
5	`void`	`multiplyDouble(double c)` Multiplies the value of `c` by the fraction. Does not change the state of the fraction.

(a) [2pt] What does the term *method overloading* mean for a class?

A class can have multiple methods with the same name as long as the method signatures are unique.

(b) [4pt] Identify which version of `multiply` is invoked for each of the Java statements shown below. Write your answer above the line provided beside each invocation of `multiply`. You may use the numbers 1–3 to identify the method version; use the letter C to identify a compilation error.

```
BetterFraction f = new BetterFraction(1, 1);


f.multiply(3.0);                  ____2_____

f.multiply(f);                    ____3_____

f.multiply(f.doubleValue());      ____2_____

f.multiply(1 / 2);                ____1_____
```

(c) [2pt]

Like the `Fraction` class, `BetterFraction` has a `public static boolean` field `isQuoted`. Show how the client can directly set this field to `false` using a single line of Java code. Use proper convention for full credit.

```
BetterFraction.isQuoted = false;
```

(d) [2pt] What does the following snippet of code print?

```
BetterFraction b = new BetterFraction(10, 1);   // fraction 10/1
double c = 3.0;
b.multiplyDouble(c);
System.out.println(c);
```

`3.0` (a method in Java cannot change the value of a primitve argument because the argument is passed by value)

4. **Objects.** [10pt]

Consider the (fictional) class `Book` that represents a printed book. Its partial API is shown below.

	Field Summary	
1	`static short`	`NUMBER_OF_BOOKS` A count of the number of books that have been created using the `Book` constructor.

	Constructor Summary	
2		`Book(String title)` Creates a book with the given title.

	Method Summary	
3	`int`	`getPages()` Returns the number of pages in the book; does not modify the book.
4	`String`	`getTitle()` Returns the title of the book; does not modify the book.
5	`AudioBook`	`makeAudioBook()` Returns an audio book version of the book; does not modify the book.
6	`void`	`setTitle(String title)` Changes the title of the book to the given title.
7	`void`	`setPages(int pages)` Changes the number of pages in the book to the given number of pages, removing pages if necessary.
8	`boolean`	`equals(Object anObject)` Compares this book to the specified object.

(a) [8pt] Consider the partial API and the following `main` method:

```
Book a = new Book("Fun with Frogs");
Book b = new Book("Gaga over Gorillas");
Book c = a;
Book d = b;
```

Fill in the blanks:

There are [2] `Book` objects are in memory.

How many copies of the field `NUMBER_OF_BOOKS` are in memory? [1]

The value of `a == b` is [false].

The value of `a.equals(d)` is [false].

A client uses [==]. to determine if two variables refer to the same object.

A client uses [equals]. to determine if two variables refer to objects with the same state.

The methods numbered [3, 4, 5]. are accessor methods. (method 5 is optional)

The methods numbered [6, 7]. are mutator methods.

(b) [2pt] Give two reasons why the attribute `NUMBER_OF_BOOKS` should not have been made `public`.

Try to provide an example using a simple Java statement for each reason, if you can.

Several possible answers:

breaks encapsulation / client can set an illegal state	`Book.NUMBER_OF_BOOKS = -1;`
name is a permanent part of the API	
type is a permanent part of the API	

5. **Control.** [10pt]

Pick *one* best answer for each. Each question is worth two points.

`Fraction` in the code snippets is from `type.lib`. Recall `Fraction`'s `compareTo` returns `-1` if the argument is larger than the fraction, `0` if they are the same, and `1` if the argument is smaller.

(a)
```
int x = 1;
int y = 3;
int z = 2;
if ((x > y) && (x > z))
{
    System.out.println("X!");
} else if ((y > x) && (y > z))
{
    System.out.println("Y!");
} else if ((z > x) && (z > y))
{
    System.out.println("Z!");
} else
{
    System.out.println("None.");
}
```

A. `X!`
B. `Y!`
C. `Z!`
D. `None.`
E. *Error.*

(b)
```
final Fraction ONE_HALF  = new Fraction(1, 2);
final Fraction ONE_EIGHT = new Fraction(1, 8);
Fraction frac = new Fraction(1, 1);
int counter = 0;
do
{
    frac.multiply(ONE_HALF);
    counter++;
} while (frac.compareTo(ONE_EIGHT) >= 0);
System.out.println(counter);
```

A. `1`
B. `3`
C. `4`
D. *Infinite loop.*
E. *Error.*

(c)
```
final Fraction ONE_HALF  = new Fraction(1, 2);
final Fraction ONE_EIGHT = new Fraction(1, 8);
Fraction frac = new Fraction(1, 1);
int counter = 0;
while (frac != ONE_EIGHT)
{
    frac.multiply(ONE_HALF);
    counter++;
}
System.out.println(counter);
```

A. 1
B. 2
C. 3
D. *Infinite loop.*
E. *Error.*

(d)
```
int j = 0;
for (int i = 1; i <= 10; i++)
{
    i++;
    System.out.print("x");
}
System.out.println("");
```

A. x
B. xxxxx
C. xxxxxxxxx
D. xxxxxxxxxx
E. *Error.*

(e)
```
int count = 0;
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < i; j++)
    {
        count++;
    }
}
System.out.println(count);
```

A. 1
B. 3
C. 6
D. *Infinite loop.*
E. *Error.*

Blank Page.