MULTIPLE-CHOICE QUESTIONS ON CLASSES AND OBJECTS

Questions 1–3 refer to the `Time` class declared below:

```
public class Time
{
    private int hrs;
    private int mins;
    private int secs;

    public Time()
    { /* implementation not shown */ }

    public Time(int h, int m, int s)
    { /* implementation not shown */ }

    /** Resets time to hrs = h, mins = m, secs = s. */
    public void resetTime(int h, int m, int s)
    { /* implementation not shown */ }

    /** Advances time by one second. */
    public void increment()
    { /* implementation not shown */ }

    /** @return true if this time equals t, false otherwise */
    public boolean equals(Time t)
    { /* implementation not shown */ }

    /** @return true if this time is earlier than t, false otherwise */
    public boolean lessThan(Time t)
    { /* implementation not shown */ }

    /** @return a String with the time in the form hrs:mins:secs */
    public String toString()
    { /* implementation not shown */ }
}
```

1. Which of the following is a *false* statement about the methods?
   (A) `equals`, `lessThan`, and `toString` are all accessor methods.
   (B) `increment` is a mutator method.
   (C) `Time()` is the default constructor.
   (D) The `Time` class has three constructors.
   (E) There are no static methods in this class.

Questions 5–11 refer to the following Date class declaration:

```
public class Date
{
    private int day;
    private int month;
    private int year;

    public Date()                                  //default constructor
    {
        ...
    }

    public Date(int mo, int da, int yr)    //constructor
    {
        ...
    }

    public int month()       //returns month of Date
    {
        ...
    }

    public int day()         //returns day of Date
    {
        ...
    }

    public int year()        //returns year of Date
    {
        ...
    }

    //Returns String representation of Date as "m/d/y", e.g. 4/18/1985.
    public String toString()
    {
        ...
    }
}
```

5. Which of the following correctly constructs a Date object in a client class?

   (A) `Date d = new (2, 13, 1947);`

   (B) `Date d = new Date(2, 13, 1947);`

   (C)
   ```
   Date d;
   d = new (2, 13, 1947);
   ```

   (D)
   ```
   Date d;
   d = Date(2, 13, 1947);
   ```

   (E) `Date d = Date(2, 13, 1947);`

9. Here is a client program that uses Date objects:

```
public class BirthdayStuff
{
    public static Date findBirthdate()
    {
        /* code to get birthDate */
        return birthDate;
    }

    public static void main(String[] args)
    {
        Date d = findBirthdate();
            ...
    }
}
```

Which of the following is a correct replacement for
/* *code to get* birthDate */?

I
```
System.out.println("Enter birthdate: mo, day, yr: ");
int m = IO.readInt();                    //read user input
int d = IO.readInt();                    //read user input
int y = IO.readInt();                    //read user input
Date birthDate = new Date(m, d, y);
```

II
```
System.out.println("Enter birthdate: mo, day, yr: ");
int birthDate.month() = IO.readInt();      //read user input
int birthDate.day() = IO.readInt();        //read user input
int birthDate.year() = IO.readInt();       //read user input
Date birthDate = new Date(birthDate.month(), birthDate.day(),
    birthDate.year());
```

III
```
System.out.println("Enter birthdate: mo, day, yr: ");
int birthDate.month = IO.readInt();      //read user input
int birthDate.day = IO.readInt();        //read user input
int birthDate.year = IO.readInt();       //read user input
Date birthDate = new Date(birthDate.month, birthDate.day,
    birthDate.year);
```

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) I and III only

12. Here are the private instance variables for a `Frog` object:

```
public class Frog
{
    private String species;
    private int age;
    private double weight;
    private Position position;      //position (x,y) in pond
    private boolean amAlive;
        ...
```

Which of the following methods in the `Frog` class is the best candidate for being a static method?

(A) `swim` `//frog swims to new position in pond`

(B) `getPondTemperature` `//returns temperature of pond`

(C) `eat` `//frog eats and gains weight`

(D) `getWeight` `//returns weight of frog`

(E) `die` `//frog dies with some probability based`
`//on frog's age and pond temperature`

13. What output will be produced by this program?

```
public class Mystery
{
    public static void strangeMethod(int x, int y)
    {
        x += y;
        y *= x;
        System.out.println(x + " " + y);
    }

    public static void main(String[] args)
    {
        int a = 6, b = 3;
        strangeMethod(a, b);
        System.out.println(a + " " + b);
    }
}
```

(A) 36
9

(B) 3 6
9

(C) 9 27
9 27

(D) 6 3
9 27

(E) 9 27
6 3

15. The constructors in the `Rational` class allow initialization of `Rational` objects in several different ways. Which of the following will cause an error?

(A) `Rational r1 = new Rational();`
(B) `Rational r2 = r1;`
(C) `Rational r3 = new Rational(2,-3);`
(D) `Rational r4 = new Rational(3.5);`
(E) `Rational r5 = new Rational(10);`

16. Here is the implementation code for the `plus` method:

```
/** Returns (this + r). Leaves this unchanged.
 *  @return this rational number plus r
 *  @param r a rational number to be added to this Rational
 */
public Rational plus(Rational r)
{
    fixSigns();
    r.fixSigns();
    int denom = denominator * r.denominator;
    int numer = numerator * r.denominator
              + r.numerator * denominator;
    /* more code */
}
```

Which of the following is a correct replacement for `/* more code */`?

(A)
```
Rational rat(numer, denom);
rat.reduce();
return rat;
```

(B)
```
return new Rational(numer, denom);
```

(C)
```
reduce();
Rational rat = new Rational(numer, denom);
return rat;
```

(D)
```
Rational rat = new Rational(numer, denom);
Rational.reduce();
return rat;
```

(E)
```
Rational rat = new Rational(numer, denom);
rat.reduce();
return rat;
```

17. Assume these declarations:

```
Rational a = new Rational();
Rational r = new Rational(numer, denom);
int n = value;
//numer, denom, and value are valid integer values
```

Which of the following will cause a compile-time error?

(A) `r = a.plus(r);`
(B) `a = r.plus(new Rational(n));`
(C) `r = r.plus(r);`
(D) `a = n.plus(r);`
(E) `r = r.plus(new Rational(n));`

18. A client method contains this code segment:

```
Temperature t1 = new Temperature(40, "C");
Temperature t2 = t1;
Temperature t3 = t2.lower(20);
Temperature t4 = t1.toFahrenheit();
```

Which statement is *true* following execution of this segment?

(A) `t1`, `t2`, `t3`, and `t4` all represent the identical temperature, in degrees Celsius.
(B) `t1`, `t2`, `t3`, and `t4` all represent the identical temperature, in degrees Fahrenheit.
(C) `t4` represents a Fahrenheit temperature, while `t1`, `t2`, and `t3` all represent degrees Celsius.
(D) `t1` and `t2` refer to the same `Temperature` object; `t3` refers to a `Temperature` object that is 20 degrees lower than `t1` and `t2`, while `t4` refers to an object that is `t1` converted to Fahrenheit.
(E) A `NullPointerException` was thrown.

19. Consider the following code:

```
public class TempTest
{
    public static void main(String[] args)
    {
        System.out.println("Enter temperature scale: ");
        String tempScale = IO.readString();     //read user input
        System.out.println("Enter number of degrees: ");
        double tempDegrees = IO.readDouble();   //read user input
        /* code to construct a valid temperature from user input */
    }
}
```

Which is a correct replacement for `/* code to construct... */`?

I
```
Temperature t = new Temperature(tempDegrees, tempScale);
if (!t.isValidTemp(tempDegrees,tempScale))
    /* error message and exit program */
```

II
```
if (isValidTemp(tempDegrees,tempScale))
    Temperature t = new Temperature(tempDegrees, tempScale);
else
    /* error message and exit program */
```

III
```
if (Temperature.isValidTemp(tempDegrees,tempScale))
    Temperature t = new Temperature(tempDegrees, tempScale);
else
    /* error message and exit program */
```

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) I and III only

21. Consider this program:

```
public class CountStuff
{
    public static void doSomething()
    {
        int count = 0;
            ...
        //code to do something - no screen output produced
        count++;
    }

    public static void main(String[] args)
    {
        int count = 0;
        System.out.println("How many iterations?");
        int n = IO.readInt();      //read user input
        for (int  i = 1; i <= n; i++)
        {
            doSomething();
            System.out.println(count);
        }
    }
}
```

If the input value for n is 3, what screen output will this program subsequently produce?

(A)
```
0
0
0
```

(B)
```
1
2
3
```

(C)
```
3
3
3
```

(D)
```
?
?
?
```
where ? is some undefined value.

(E) No output will be produced.

23. Consider the following program:

```
public class Tester
{
    public void someMethod(int a, int b)
    {
        int temp = a;
        a = b;
        b = temp;
    }
}

public class TesterMain
{
    public static void main(String[] args)
    {
        int x = 6, y = 8;
        Tester tester = new Tester();
        tester.someMethod(x, y);
    }
}
```

Just before the end of execution of this program, what are the values of `x`, `y`, and `temp`, respectively?

(A) 6, 8, 6
(B) 8, 6, 6
(C) 6, 8, ?, where ? means undefined
(D) 8, 6, ?, where ? means undefined
(E) 8, 6, 8