MULTIPLE-CHOICE QUESTIONS ON CLASSES AND OBJECTS

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

```
public class Time
{
    private int myHrs;
    private int myMins;
    private int mySecs;

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

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

    //Resets time to myHrs = h, myMins = m, mySecs = s.
    public void resetTime(int h, int m, int s)
    { /* implementation not shown */ }

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

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

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

    //Returns time as a String 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.

2. Which of the following represents correct *implementation code* for the constructor with parameters?

(A)
```
myHrs = 0;
myMins = 0;
mySecs = 0;
```

(B)
```
myHrs = h;
myMins = m;
mySecs = s;
```

(C)
```
resetTime(myHrs, myMins, mySecs);
```

(D)
```
h = myHrs;
m = myMins;
s = mySecs;
```

(E)
```
Time = new Time(h, m, s);
```

3. A client class has a `display` method that writes the time represented by its parameter:

```
//Outputs time t in the form hrs:mins:secs.
public void display (Time t)
{
    /* method body */
}
```

Which of the following are correct replacements for `/* method body */`?

I
```
Time T = new Time(h, m, s);
System.out.println(T);
```

II
```
System.out.println(t.myHrs + ":" + t.myMins + ":" + t.mySecs);
```

III
```
System.out.println(t);
```

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

4. Which statement about parameters is *false*?
(A) The scope of parameters is the method in which they are defined.
(B) Static methods have no implicit parameter `this`.
(C) Two overloaded methods in the same class must have parameters with different names.
(D) All parameters in Java are passed by value.
(E) Two different constructors in a given class can have the same number of parameters.

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

```
public class Date
{
    private int myDay;
    private int myMonth;
    private int myYear;

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

    public Date(int mo, int day, 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?

   (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);
   ```

6. Which of the following will cause an error message?

I
```
Date d1 = new Date(8, 2, 1947);
Date d2 = d1;
```

II
```
Date d1 = null;
Date d2 = d1;
```

III
```
Date d = null;
int x = d.year();
```

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

7. A client program creates a `Date` object as follows:

```
Date d = new Date(1, 13, 2002);
```

Which of the following subsequent code segments will cause an error?
(A) `String s = d.toString();`
(B) `int x = d.day();`
(C) `Date e = d;`
(D) `Date e = new Date(1, 13, 2002);`
(E) `int y = d.myYear;`

8. Consider the implementation of a `write()` method that is added to the `Date` class:

```
//Write the date in the form m/d/y, for example 2/17/1948.
public void write()
{
    /* implementation code */
}
```

Which of the following could be used as `/* implementation code */`?

I
```
System.out.println(myMonth + "/" + myDay + "/" + myYear);
```

II
```
System.out.println(month() + "/" + day() + "/" + year());
```

III
```
System.out.println(this);
```

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

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.myMonth = IO.readInt();       //read user input
int birthDate.myDay = IO.readInt();         //read user input
int birthDate.myYear = IO.readInt();        //read user input
Date birthDate = new Date(birthDate.myMonth, birthDate.myDay,
    birthDate.myYear);
```

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

10. A method in a client program for the `Date` class has this declaration:

```
Date d1 = new Date(month, day, year);
```

where `month`, `day`, and `year` are previously defined integer variables. The same method now creates a second `Date` object `d2` that is an exact copy of the object `d1` refers to. Which of the following code segments will *not* do this correctly?

I
```
Date d2 = d1;
```

II
```
Date d2 = new Date(month, day, year);
```

III
```
Date d2 = new Date(d1.month(), d1.day(), d1.year());
```

(A) I only
(B) II only
(C) III only
(D) I, II, and III
(E) All will do this correctly.

11. The `Date` class is modified by adding the following mutator method:

```
public void addYears(int n)    //add n years to date
```

Here is part of a poorly coded client program that uses the `Date` class:

```
public static void addCentury(Date recent, Date old)
{
    old.addYears(100);
    recent = old;
}

public static void main(String[] args)
{
    Date oldDate = new Date(1, 13, 1900);
    Date recentDate = null;
    addCentury(recentDate, oldDate);
        ...
}
```

Which will be true after executing this code?
(A) A `NullPointerException` is thrown.
(B) The `oldDate` object remains unchanged.
(C) `recentDate` is a null reference.
(D) `recentDate` refers to the same object as `oldDate`.
(E) `recentDate` refers to a separate object whose contents are the same as those of `oldDate`.

Questions 12–15 refer to the following definition of the `Rational` class:

```
public class Rational
{
    private int myNum;          //numerator
    private int myDenom;        //denominator

    //constructors
    /* default constructor */
    Rational()
    { /* implementation not shown */ }

    /* Constructs a Rational with numerator n and
     * denominator 1. */
    Rational(int n)
    { /* implementation not shown */ }

    /* Constructs a Rational with specified numerator and
     * denominator. */
    Rational(int numer, int denom)
    { /* implementation not shown */ }

    //accessors
    /* Returns numerator. */
    int numerator()
    { /* implementation not shown */ }

    /* Returns denominator. */
    int denominator()
    { /* implementation not shown */ }

    //arithmetic operations
    /* Returns (this + r).
     * Leaves this unchanged. */
    public Rational plus(Rational r)
    { /* implementation not shown */ }

    //Similarly for times, minus, divide
          ...
    /* Ensures myDenom > 0. */
    private void fixSigns()
    { /* implementation not shown */ }

    /* Ensures lowest terms. */
    private void reduce()
    { /* implementation not shown */ }
}
```

12. The method `reduce()` is not a public method because
    (A) methods whose return type is `void` cannot be public.
    (B) methods that change `this` cannot be public.
    (C) the `reduce()` method is not intended for use by clients of the `Rational` class.
    (D) the `reduce()` method is intended for use only by clients of the `Rational` class.
    (E) the `reduce()` method uses only the private data fields of the `Rational` class.

13. 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);`

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

```
/* Returns (this + r) in reduced form. Leaves this unchanged. */
public Rational plus(Rational r)
{
    fixSigns();
    r.fixSigns();
    int denom = myDenom * r.myDenom;
    int num = myNum * r.myDenom + r.myNum * myDenom;
    /* some more code */
}
```

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

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

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

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

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

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

15. Assume these declarations:

```
Rational a = new Rational();
Rational r = new Rational(num, denom);
int n = value;
//num, 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));`

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

```
public class Frog
{
    private String mySpecies;
    private int myAge;
    private double myWeight;
    private Position myPosition;      //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`

17. 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

Questions 18–20 refer to the Temperature class shown below:

```
public class Temperature
{
    private String myScale;  //valid values are "F" or "C"
    private double myDegrees;

    //constructors
    /* default constructor */
    public Temperature()
    { /* implementation not shown */ }

    /* constructor with specified degrees and scale */
    public Temperature(double degrees, String scale)
    { /* implementation not shown */ }

    //accessors
    /* Returns degrees for this temperature. */
    public double getDegrees()
    { /* implementation not shown */ }

    /* Returns scale for this temperature. */
    public String getScale()
    { /* implementation not shown */ }

    //mutators
    /* Precondition:  Temperature is a valid temperature
     *                in degrees Celsius.
     * Postcondition: Returns this temperature, which has been
     *                converted to degrees Fahrenheit. */
    public Temperature toFahrenheit()
    { /* implementation not shown */ }

    /* Precondition:  Temperature is a valid temperature
     *                in degrees Fahrenheit.
     * Postcondition: Returns this temperature, which has been
     *                converted to degrees Celsius. */
    public Temperature toCelsius()
    { /* implementation not shown */ }

    /* Raise this temperature by amt degrees and return it. */
    public Temperature raise(double amt)
    { /* implementation not shown */ }

    /* Lower this temperature by amt degrees and return it. */
    public Temperature lower(double amt)
    { /* implementation not shown */ }

    /* Returns true if the number of degrees is a valid
     * temperature in the given scale, false otherwise. */
    public static boolean isValidTemp(double degrees, String scale)
    { /* implementation not shown */ }

    //other methods not shown ...
}
```

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 scale = IO.readString();     //read user input
        System.out.println("Enter number of degrees: ");
        double degrees = 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(degrees, scale);
if (!t.isValidTemp(degrees,scale))
    /* error message and exit program */
```

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

III
```
if (Temperature.isValidTemp(degrees,scale))
    Temperature t = new Temperature(degrees, scale);
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

20. The formula to convert degrees Celsius C to Fahrenheit F is

$$F = 1.8C + 32$$

For example, 30° C is equivalent to 86° F.

An `inFahrenheit()` accessor method is added to the `Temperature` class. Here is its implementation:

```
/* Precondition:  temperature is a valid temperature in
 *                degrees Celsius
 * Postcondition: an equivalent temperature in degrees
 *                Fahrenheit has been returned. Original
 *                temperature remains unchanged */
public Temperature inFahrenheit()
{
    Temperature result;
    /* more code */
    return result;
}
```

Which of the following correctly replaces `/* more code */` so that the postcondition is achieved?

I
```
result = new Temperature(myDegrees*1.8 + 32, "F");
```

II
```
result = new Temperature(myDegrees*1.8, "F");
result = result.raise(32);
```

III
```
myDegrees *= 1.8;
this = this.raise(32);
result = new Temperature(myDegrees, "F");
```

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

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.

22. This question refers to the following class:

```
public class IntObject
{
    private int myInt;

    public IntObject()        //default constructor
    { myInt = 0; }
    public IntObject(int n)   //constructor
    { myInt = n; }
    public void increment()   //increment by 1
    { myInt++; }
}
```

Here is a client program that uses this class:

```
public class IntObjectTest
{
    public static IntObject someMethod(IntObject obj)
    {
        IntObject ans = obj;
        ans.increment();
        return ans;
    }

    public static void main(String[] args)
    {
        IntObject x = new IntObject(2);
        IntObject y = new IntObject(7);
        IntObject a = y;
        x = someMethod(y);
        a = someMethod(x);
    }
}
```

Just before exiting this program, what are the object values of x, y, and a, respectively?

(A) 9, 9, 9
(B) 2, 9, 9
(C) 2, 8, 9
(D) 3, 8, 9
(E) 7, 8, 9

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