Methods and Classes Test 2

1. In baseball a player’s batting average is calculated by dividing the number of hits by the number of times the player has batted. Assume the following class has been defined.

public class Baseball

{

/\* precondition: atBats > 0

\* postcondition: a batting average has been computed

\* @ param hits the number of hits

\* @ param atBats the number of atBats

\* @ return the batting average

\*/

public static double **average**(int hits, int atBats)

{

// implementation not shown

}

// other methods not shown

}

Which of the following best demonstrates the use of method average as a static method when it is called from   
 a method in another class?

1. player.average();
2. average(100, 300);
3. baseball.average(100, 300);
4. Baseball.average(100, 300);
5. The area of a rectangle is calculated by multiplying the length times the width. Assume the following class has been defined.

public class Rectangle

{

\* postcondition: the area has been computed

\* @ return the area

\*/

public double **area**()

{

// implementation not shown

}

// other methods not shown

}

Given the following definition:

Rectangle rect = new Rectangle();

Which of the following lines of code will correctly calculate the area of a rectangle?

1. Rectangle.area();
2. rect.area(length, width);
3. area();
4. rect.area();
5. Look at the following method declaration

public \_\_\_\_\_\_\_\_\_\_\_\_\_\_ sum(double num1, double num2) {  
 return num1 + num2;  
}

Which of the following can be inserted into the blank so that the method will correctly  
 calculate and return the sum of num1 and num2?

1. static
2. double
3. void
4. int
5. Look at the following method declaration

public void play(int a)

Which of the following will **not** correctly overload the method **play**?

1. public void play(double a)
2. public void play(int a, int b)
3. public int play(int a)
4. public void play()
5. Look at the following code segment.

public class Demo {  
 public static void go(int num) {  
 num = num + 100;  
 }

public static void main(String[] args) {  
 int number = 25;  
 go(number);  
 System.out.println(number);  
 }

}

What would be output by the code above?

1. 125
2. 100
3. 25
4. No output due to a run-time error

Questions 6-10 involve questions related to the following class.

public class Fraction

{

private int num; // numerator

private int denom; // denominator

public Fraction(int n, int d)

{

num = n;

denom = d;

}

public int getNum()

{

return num;

}

public int getDenom()

{

return denom;

}

public void setNum(int n)

{

num = n;

}

public void setDenom(int d)

{

denom = d;

}

public String toString()

{

return num + "/" + denom;

}

}

1. Which of the following will correctly instantiate a Fraction object?
2. num = 3;

denom = 4;

1. Fraction f = new Fraction(3, 4);
2. Fraction f = new Fraction();
3. Fraction.num = 3;

Fraction.denom = 4;

1. Assume a fraction object has been created and its reference is stored in a variable named frac. Which of the following will correctly modify the numerator value of a fraction?
2. frac.getNum() = 5;
3. num = 5;
4. frac.setNum(5);
5. Fraction.num = 5;
6. Assume a fraction object has been created and its reference is stored in a variable named frac. Which of the following will not print frac in the form 3/4?
7. System.out.println(frac);
8. System.out.println(frac.toString());
9. System.out.println(num + "/" + denom);
10. System.out.println(frac.getNum() + "/" + frac.getDenom());
11. Suppose the following **equals** method is added to the Fraction class.

public boolean equals(Fraction other)  
{  
 return /\* code implementation \*/  
}

Which of the following lines will **not** correctly replace /\* code implementation \*/ so that   
 two fractions can be compared for equality?

1. getNum() == num && getDenom() == denom;
2. num == other.num && denom == other.denom;
3. this.num == other.num && this.denom == other.denom;
4. this.getNum() == other.getNum() && this.getDenom() == other.getDenom();

10. Which line contains a syntax error?

Line 1 : Fraction frac = new Fraction(3, 4);  
Line 2 : frac.setNum(3);

Line 3 : frac.setDenom(frac.getNum());

Line 4 : System.out.println(getNum() + "/" + getDenom());

1. Line 1
2. Line 2
3. Line 3
4. Line 4

**Free Response**

1. Write a complete class definition for a class named Vehicle that stores information about an automobile. The class should include the following:

* Three **instance** **variables** representing the make, model, and year of the vehicle.
* Two **constructors** - a no parameter (default) constructor and a three parameter constructor that will initialize the three instance variables.
* An **accessor** method for each instance variable.
* A **mutator** method for each instance variable.
* A **toString** method that will label and display each instance variable’s value.