

1. Consider the following class:

```
public class Fraction implements Comparable
{
    public int getNum() { return num; }
    public int getDenom() { return denom; }
    public double doubleValue() { return (double) num / denom; }
    < other constructors and methods not shown >
    private int num, denom;
}
```

Which of the following would appropriately implement a compareTo method, required by the Comparable interface?

I.
public int compareTo(Fraction other)
{
 return getNum() * other.getDenom() - getDenom() * other.getNum();
}

II.
public int compareTo(Object other)
{
 double x = doubleValue();
 double y = ((Fraction) other).doubleValue();
 if (x < y) return -1;
 else if (x > y) return 1;
 else return 0;
}

III.
public int compareTo(Object other)
{
 return (int) (doubleValue() - ((Fraction) other).doubleValue());
}

- A. I only
- B. II only
- C. I and II
- D. II and III
- E. I, II, and III

2. What is the output from the following code segment?

```
Comparable x = new Integer(123);           // Line 1
System.out.println(x.compareTo("123"));    // Line 2
```

- A. 0
- B. a positive integer
- C. a syntax error on Line 1
- D. a syntax error on Line 2
- E. A ClassCastException

3. How many interfaces can a given class implement?

- A. none
- B. 1
- C. 2
- D. as many as it needs to

4. How many classes can a given interface implement?

- A. none
- B. 1
- C. 2
- D. as many as it needs to

5. Which of the following statements is true about the `compareTo` method?

- I. `compareTo` can be called by any object
- II. `compareTo` can be redefined by any class
- III. `compareTo` can be implemented by any class that implements the `Comparable` interface

- A. I only
- B. II only
- C. III only
- D. I and II
- E. none are true

6. Which of the following classes is the least suitable candidate for implementing the `Comparable` interface?

A.

```
public class Point
{
    private double x;
    private double y;
    // other methods
}
```

B.

```
public class Name
{
    private String firstName;
    private String lastName;
    // other methods
}
```

C.

```
public class Car
{
    private int modelNumber;
    private int year;
    private double price;
    // other methods
}
```

D.

```
public class Student
{
    private String name;
    private double gpa;
    // other methods
}
```

7. Consider the Temperature class defined below:

```
public class Temperature implements Comparable
{
    private String myScale;
    private double myDegrees;

    public Temperature () { implementation code }
    public Temperature(String scale, double degrees){ implementation code }
    public int compareTo(Object obj) { implementation code }
    public String toString() { implementation code }
}
```

Here is a program that finds the lowest of three temperatures:

```
public class TemperatureMain
{
    // find smaller of objects a and b
    public static Comparable min(Comparable a, Comparable b)
    {
        if (a.compareTo(b) < 0)
            return a;
        else
            return b;
    }

    // find smallest of objects a, b, and c
    public static Comparable minThree(Comparable a, Comparable b, Comparable c)
    {
        return min(min(a, b), c);
    }

    public static void main(String[] args)
    {
        < code to test minThree method >
    }
}
```

Which are correct replacements for <code to test minThree method>?

I.
Temperature t1 = new Temperature("C", 85);
Temperature t2 = new Temperature("F", 45);
Temperature t3 = new Temperature("F", 120);
System.out.println("The lowest temperature is " + minThree(t1, t2, t3));

II.
Comparable c1 = new Temperature("C", 85);
Comparable c2 = new Temperature("F", 45);
Comparable c3 = new Temperature("F", 120);
System.out.println("The lowest temperature is " + minThree(c1, c2, c3));

III.
Comparable c1 = new Comparable ("C", 85);
Comparable c2 = new Comparable ("F", 45);
Comparable c3 = new Comparable ("F", 120);
System.out.println("The lowest temperature is " + minThree(c1, c2, c3));

- A. II only
- B. I and II only
- C. II and III only
- D. I and III only
- E. I, II, and III

8. Consider the following interface and class definitions:

```
public interface Employee
{
    void raiseSalary();
}

public interface Musician
{
    void play();
}

public class Test implements Employee, Musician
{
    public void raiseSalary()
    {
        System.out.println("raising");
    }

    public void play()
    {
        System.out.println("playing");
    }
}
```

Which of the following statements about these definitions is true?

- A. The code will not compile because class `Test` tries to implement the two interfaces at once.
- B. The code will not compile because class `Test` has no properties.
- C. The code will not compile because class `Test` only implements the methods defined in the `Employee` and `Musician` classes; it does not define any new methods;
- D. The code will compile; however, if class `Test` did not include a definition of the `play` method, the code would not compile.
- E. The code will compile; furthermore, even if class `Test` did not include a definition of the `play` method, the code would compile.