Instructions: closed books, closed notes, open minds,
1 hour 15 minute time limit.
There are 4 sections for a total of 80-ish points:

Part I:  20-ish points
Part II: 12-ish points
Part III: 37-ish points
Part IV:  8-ish points
Part I  (20 points total)

1. Each of the following contains an error that will cause it to not compile (a syntax error) or to behave other than as expected (a logic error). For each, describe the error.  (1 point each)

   a) int i = 3.14;
      syntax: 3.14 not an int value

   b) double f = 3.6
      syntax: no ;

   c) while ( k < 3 );
      logic: empty loop body thanks to ; after while

   d) for(double d = 0.0, d < 3.0, d++)
      System.out.print(d + " ");
      syntax: commas instead of ;

   e) for (int k=0; k < 20; k++)
      System.out.println(k);
      System.out.println(k);
      syntax: k not defined outside loop

   f) if ( colorint = 0 )
      fishColor = Color.red;
      logic: single = means assignment

2. Fully parenthesize the following expression to indicate the order of evaluation.  (2 pts)

   (((x * y) + a) - (b / c))
3. For each of the following expressions, tell the **type** of the expression, along with its **value**.
   (2 points each)

   Assume the following variable definitions and function declaration:
   ```
   int m = 2;  // m is an int
   int pi = (int) 3.14159;  // pi is an int
   double c = 3.0, b = 3.0, h = 4.0; // c, b, and h are doubles
   double sqrt(double sqr);  // returns the square root of
   ```

   ```
   a) m * c * c
       double 18.0
   
   b) sqrt(b*b + h*h)
       double 5.0
   
   c) pi
       int 3
   ```

4. Give logical (i.e. Boolean) expressions that represent the following phrases. (2 points each)
   a) i and j are greater than 10
      ```
      i > 10 && j > 10
      ```
   
   b) i is positive or less than -10
      ```
      i > 0 || i < -10
      ```
   
   c) i is less than j, which is less than k
      ```
      i < j && j < k
      ```
Part II. (20 points total)

5. Consider the following code segment that is intended to print different welcome messages at different times of the day. (6 points)

```java
if ( hour < 12 )
    System.out.println("Good morning.");
if ( hour < 18 )
    System.out.println("Good afternoon.");
if ( hour < 24 )
    System.out.println("Good night.");
```

a) What is the actual behavior of this code segment?

It prints all three messages between midnight and noon, the second two messages between noon and 6pm, and a single message (“Good night”) after 6pm.

b) How would you modify this code segment so that its actual behavior matches its intended behavior? (you may make changes by editing the segment above or by writing a version of your own below)

There are 2 approaches: could use “else if” for the second “if” and an “else” (with or without the “if ( hour < 24 )”) at the end, OR could change the second and third conditions to ( hour >= 12 && hour < 18 ) and ( hour >= 18 && hour < 24 ), respectively.

6. How many times is statement S executed in the following Java code fragments? (3 points each)

a) for (i = 0; i <= A.size(); i++)  
   S  
   b) for (i = 1; i < N; i++)  
      for (j = 0; j < M; j++)  
         S  

A.size() + 1 times  
(N-1) * M times
Various questions throughout this exam will refer to the classes whose instance variables and method signatures are shown below. (The method bodies are not shown.)

```java
public class CDInfo
{
    // instance variables
    private String title;
    private String artist;
    private ArrayList<String> songTItles;
    private double price;

    // constructor
    public CDInfo(String cdTitle, String cdArtistName,
                   ArrayList<String> cdSongs, double cdPrice) {...}

    // observer or accessor methods
    public String getTitle () {...} // title of CD
    public String getArtistName() {...} // name of artist
    public ArrayList<String> getSongs() {...} // list of songs
    public double getPrice(){...} // price of CD
}

public class CDDatabase
{
    // instance variables
    private ArrayList<CDInfo> cdList;

    // constructors not shown

    // observer or accessor methods
    public void printAll() {...}
    public int numTitlesBy(String artistName) {...}
    public double costOf(String cdTitle) {...}
    public String cheapestCD(){...}
    public void printSongTitles(String cdTitle) {...}
    public ArrayList<CDInfo> allCDsLessThan(double price) {...}

    // other methods not shown
}
```
Part III. (28 points total)

7. Write the code for the CDInfo constructor. (3 points)

```java
public CDInfo(String cdTitle, String cdArtistName,
               ArrayList<String> cdSongs, double cdPrice)
{
    this.title = cdTitle; // this. is optional
    this.artist = cdArtistName,
    this.songTitles = cdSongs;
    this.price = cdPrice;
}
```

8. Write the code to declare a variable that refers to a CDInfo object and construct and initialize the object with the following information: (3 points)
title: “Rubber Soul”
artist: “Beatles”
songTitles: assume that the song titles are contained in an ArrayList called songList
price: 12.95

```java
CDInfo myCD = new CDInfo(“Rubber Soul”, “Beatles”, songList, 12.95);
```

9. Write the code to print the title of the CDInfo object you constructed above, using the appropriate method (i.e., do not just print “Rubber Soul”). (2 points)

```java
System.out.println(myCD.getTitle());
```

10. Write the code for the CDDatabase method printAll, which prints the artist name, followed by the title and price, for all CDs in the database. (Note that this function does not print the song titles.) Use spaces and line breaks to make your output legible. (4 points)

```java
public void printAll()
{
    for (CDInfo cd : this.cdList)
    {
        System.out.println(cd.getArtistName() + " : " +
                            cd.getTitle() + " : " +
                            cd.getPrice());
    }
    /* OR for (int i = 0; i < this.cdList.size(); i++)
    {
        CDInfo cd = this.cdList.get(i);
        // same System.out.println as above
    } */
}
11. Write the code for the `CDDatabase` method `numTitlesBy`, which returns the number of CDs in the database by the specified artist. (4 points)

```java
public int numTitlesBy(String artistName)
{
    int count = 0;
    for (CDInfo cd : this.cdList)
    {
        if (cd.getArtistName().equals(artistName))
            count++;
    }
    return count;
}
```

12. Write the code for the `CDDatabase` method `costOf`, which returns the price of the CD whose title is provided. If there is no CD in the database with the given title, return a price of 0.0. (4 points)

```java
public double costOf(String cdTitle)
{
    for (CDInfo cd : this.cdList)
    {
        if (cd.getTitle().equals(cdTitle))
            return cd.getPrice();
    }
    return 0.0;
}
```
13. Write the code for the CDDatabase method cheapestCD, which returns the title of the cheapest CD in the database. If there are no CDs in the database, this function should return a null string. If there is more than one CD with the lowest price, you may return any one of them (whichever you find easiest). (5 points)

```java
// There are several ways to do this, but here is one...
public String cheapestCD()
{
    if (this.cdList.size() == 0)
        return null;
    CDInfo cheapestSoFar = this.cdList.get(0);
    for (CDInfo cd : this.cdList)
    {
        if (cd.getPrice() < cheapestSoFar.getPrice())
            cheapestSoFar = cd;
    }
    return cheapestSoFar.getTitle();
}
```

14. Write the code for the CDDatabase method printSongTitles, which prints the titles of the songs of the CD whose title is provided, one song per line. (6 points)

```java
public void printSongTitles(String cdTitle)
{
    for (CDInfo cd : this.cdList)
    {
        if (cd.getTitle().equals(cdTitle))
        {
            for (String song = cd.getSongs())
                System.out.println(song);
        }
    }
    /* OR
    for (int i = 0; i < cdList.size(); i++)
    {
        CDInfo cd = cdList.get(i);
        if (cd.getTitle().equals(cdTitle))
            for (int j = 0; j < cd.getSongs().size(); j++)
            {
                System.out.println(cd.getSongs().get(j));
            }
    }*/
}
```
15. Write the code for the `CDDatabase` method `allCDsLessThan`, which returns an `ArrayList` of all the CDs in the database whose price is less than the parameter. If there are no CDs whose price is less than the parameter, return an empty `ArrayList`. (6 points)

```java
public ArrayList<CDInfo> allCDsLessThan(double price) {
    ArrayList<CDInfo> cheapCDs = new ArrayList<CDInfo>();
    for (CDInfo cd : this.cdList) {
        if (cd.getPrice() < price) {
            cheapCDs.add(cd);
        }
    }
    return cheapCDs;
}
```

Part IV. (8 points total)

The final questions in this practice exam refer to the `AquaFish` class from the Aquarium Lab Series, some of whose method signatures are shown below. (The method bodies are not shown.)

```java
public class AquaFish {
    // constructors
    public AquaFish(Aquarium aqua) {...}
    public AquaFish(Aquarium aqua, Color aColor) {...}

    // partial list of methods
    public boolean atWall() {...}  // is fish at wall?
    public void changeDir() {...}  // change direction
    public void moveForward() {...}  // move forward
    ...
}
```
16. Assume that you have an Aquarium object in a variable called `aqua`. Construct a random number generator and an empty `ArrayList` of `AquaFish`. Then construct 15 fish with random colors (each fish has a random amount of red, a random amount of green, and a random amount of blue) and add them to your new `ArrayList` and to the aquarium. (4 pts)

```java
public static void main()
{
    // There’s some code that comes before what you need to write.
    Random gen = new Random();
    ArrayList<AquaFish> fishes = new ArrayList<AquaFish>();
    for (int i = 0; i < 15; i++)
    {
        AquaFish newFish = new AquaFish(aqua,
            new Color(gen.nextInt(256), gen.nextInt(256),
                       gen.nextInt(256)));
        fishes.add(newFish);
       qua.add(newFish);
    }
}
```

17. Using the `ArrayList` from 16, write a loop to move all of the fish in your `ArrayList` forward 20 times. Each time they move, you should first to see if they are at a wall and need to turn before moving. (You do not need to add a random chance of turning if a fish is not at the wall, even though we added that functionality in the Aquarium Lab Series mini-labs.) (4 pts)

```java
for (int step = 0; step < 20; step++)
{
    for (AquaFish fish : fishes)
    {
        if (fish.atWall())
            fish.changeDir();
        fish.moveForward();
    }
    // Question didn’t ask for this, but it’s a nice touch:
    display.showAquarium();
}
```