Question 1. [4 points] Write code to prompt the user to enter her age, and then print a message indicating whether her age is even or odd.

Question 2. [4 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
public class Q2 {
  public static void f(String s) {
    s = "bar";
  }
```

```
  public static void main(
      String[] args) {
    String q = "foo";
    System.out.println(q);
    f(q);
    System.out.println(q);
  }

}
```

Question 3. [4 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
public class Q3 {
  public static void f(int[] a) {
    a = new int[2];
    a[0] = 4;
    a[1] = 5;
  }
```

```
  public static void main(
      String[] args) {
    int[] x = new int[2];
    x[0] = 2;
    x[1] = 3;
    System.out.printf("%d,%d\n",
      x[0], x[1]);
    f(x);
    System.out.printf("%d,%d\n",
      x[0], x[1]);
  }
}
```

Question 4. [4 points] Consider the following method:

```
public int countOccurrences(String fileName, char ch) throws IOException {
  FileReader r = new FileReader(fileName);
  int count = 0;
  while (true) {
    int c = r.read();
    if (c < 0) { break; }
    if (c == ch) { count++; }
  }
  r.close();
  return count;
}
```

(a) Briefly explain why this method might not attempt to close the `FileReader` even though it was created successfully.

(b) Explain how to guarantee that an attempt is made to close the `FileReader`.

Question 5. [4 points] Consider the following method:

```
public void q5(int x) {
  try {
    System.out.println("A");
    if (x < 0) {
      throw new RuntimeException();
    }
    System.out.println("B");
  } finally {
    System.out.println("C");
  }
}
```

What possible outputs can be printed when this method is executed?

Question 6. [4 points] Consider the following table (relation) names and attributes for a database containing information about movies and movie characters:

Table name	Attribute names
characters	character_id, movie_id, character_name, actor_name
movies	movie_id, movie_title, year

Modify the following query so that it returns the titles of movies in which the character Obi-Wan Kenobi appeared, along with the actor that portrayed Obi-Wan in each movie.

NOTE: `character_id` is the primary key in the `characters` table, and `movie_id` is the primary key in the `movies` table and a foreign key in the `characters` table.

```
select characters.actor_name

    from characters

    where characters.character_name = 'Obi-Wan Kenobi'
```

Question 7. [5 points] Complete the following method, called `everyN`. It takes two parameters: `list`, a `List` of elements of type `E`, and an integer `n`, which you can assume is positive. It should return a `List` of elements of type `E` containing every `n`th element starting from the first. For example, if `n` is 2, it should return a list with the first, third, fifth etc. elements. If `n` is 3, it should return the first, fourth, seventh etc. elements.

Example JUnit tests:

```
List<Integer> example = Arrays.asList(
  63, 12, 54, 82, 26, 57, 60, 80, 89, 17);

List<Integer> a = Q7.everyN(example, 3);
assertEquals(4, a.size());
assertEquals((Integer)63, a.get(0));
assertEquals((Integer)82, a.get(1));
assertEquals((Integer)60, a.get(2));
assertEquals((Integer)17, a.get(3));

List<Integer> b = Q7.everyN(example, 4);
assertEquals((Integer)3, b.size());
assertEquals((Integer)63, b.get(0));
assertEquals((Integer)26, b.get(1));
assertEquals((Integer)89, b.get(2));
```

Important: the method should not modify the contents of the `list` parameter. It should create a new empty list, add the appropriate elements, and return it.

```
public static<E> List<E> everyN(List<E> list, int n) {
```

Question 8. [4 points] State a big-O worst case upper bound on the running time of the following method, where the problem size N is the number of elements in the `ArrayList` passed as its parameter. Briefly explain your answer.

```
public static int sumOdd(ArrayList<Integer> list) {
  int sum = 0;
  for (int i = 0; i < list.size(); i++) {
    if ((list.get(i) % 2) == 1) {
       sum += list.get(i);
    }
  }
  return sum;
}
```

Question 9. [4 points] State a big-O worst case upper bound on the running time of the following method, where the problem size N is the value passed into the method as its parameter. Briefly explain your answer.

```
public static int busyWork(int N) {
  int count = 0;
  for (int i = 0; i < N * N; i++) {
    for (int j = 0; j < Math.log(N); j++) {
       count++;
    }
  }
  return count;
}
```

Note that the `Math.log` method returns the natural (base e) logarithm of its argument.

Question 10. [4 points] State a big-O worst case upper bound on the running time of the following method, where the problem size N is the number of elements in the `LinkedList` passed as its parameter. Explain your answer briefly.

```
public static int countEven(LinkedList<Integer> list) {
  int count = 0;
  Iterator<Integer> i = list.iterator();
  while (i.hasNext()) {
    Integer x = i.next();
    if (x % 2 == 0) { count++; }
  }
  return count;
}
```

Question 11. [4 points] List the 4 basic steps (in order) that are necessary to create and run `N` parallel threads for a computation that can be divided into `N` independent, non-overlapping tasks using the fork-join method. You do not need to specify the code, just the steps you would need to implement.

Question 12. [5 points] Consider the following method which computes the `n`th member of the Fibonacci sequence:

```
public static int fib(int n) { // assume n is non-negative
  if (n < 2) { return 1; }
  return fib(n-2) + fib(n-1);
}
```

Show a memoized version of this method called `fibMemo`. Assume that the `fib` function will be defined this way:

```
public static int fib(int n) { // assume n is non-negative
  return fibMemo(n, new int[n+1]);
}
```

Recall that the value 0 does not occur anywhere in the Fibonacci sequence, and that the default value for uninitialized `int` array elements is 0. Make sure that the memoization table is used to avoid computing the answer to the same subproblem multiple times.

```
public static int fibMemo(int n, int[] memo) {
```

Programming Questions

To get started, use a web browser to download the zipfile as specified by your instructor. Import it as an Eclipse project using File → Import... → General → Existing Projects into Workspace → Archive file.

Important: You may use the following resources:

- The textbook
- The lecture notes posted on the course web page
- Your previous labs and assignments

Do not open any other files, web pages, etc.

Question 13. [20 points] For the given super-class definition for `Shape`, implement the concrete sub-class `Circle`, which defines a circle shape. `Circle` must define a double named `radius` as a class field, which can only be set by being passed to the `Circle` constructor. Make sure that `shapeName` is set to "Circle" in the `Circle` constructor, and that the circumference and area fields of `Shape` can only be set through the `Circle` constructor, based on the radius value passed to the constructor. Implement a getter method for `radius`. Also, note that `Shape` implements `Comparable`, and all sub-classes of `Shape` should be compared by their respective areas.

There are unit tests in `Q13Test`: make sure they all pass. If all of the test cases pass, there is a high probability that you have correctly implemented the vast majority of the problem.

Hints:

- The `Circle` constructor should accept `radius`, set `shapeName` to "Circle", set `radius` to the value passed in, and then calculate and set `circumference` and `area` based on `radius`.
- Your implementation should only allow sub-classes to set the radius, area, and circumference values - thus you should NOT be providing any setter methods in the `Circle` class. All of the setter methods you will need are already defined in the `Shape` class.
- Look at the `Shape` class to identify which other methods you need to implement in the `Circle` class.
- As long as you have compiler errors in the test cases (red), you have additional methods to define, or you have not defined those methods as they are being used in the test cases. Thus, the problem is in your `Circle` class code. DO NOT CHANGE THE TEST CASES!!!

[Continues on other side]

Question 14. [15 points] Complete the `maxDigit` method in the `Q14` class so that it returns the value of the maximum digit in the integer `n` passed as its parameter. For example, the call

```
Q14.maxDigit(202635)
```

should return `6`.

Important: You must use recursion. Do not use a loop.

There are unit tests in `Q14Test`: make sure they pass.

Hints:

- Think of an appropriate base case
- Think of a way to identify and recursively solve a subproblem that will help you find the overall answer
- `n%10` is value of the rightmost digit of `n`
- `n/10` is an integer containing all of the digits of `n` except the rightmost digit

Question 15. [15 points] Complete the implementation of the `notInIntersection` method in the `Q15` class. It should return a `Collection` containing all elements from `a` and `b` (the two `Collection`s of elements of type `E` passed as parameters) that are not members of *both* `a` and `b`.

Requirement: The method should complete in $O(N \log N)$ time, where N is the total number of elements in `a` and `b`. (If your method has correct behavior, but does not meet this efficiency requirement, you will receive substantial partial credit.)

There are unit tests in the `Q15Test` class: make sure they pass.

Hints:

- Because `a` and `b` are `Collection`s (and not `List`s), they don't have a `get` method
- The method should not modify either `a` or `b`: it should create a new collection (e.g., `List` or `Set` containing the elements to be included in the result
- The type `E` is guaranteed to implement `Comparable`
- Sets (e.g., `HashSet` and `TreeSet`) support efficient implementations of the `add`, `remove`, and `contains` methods