Name:______________________________ NetID:____________

Answer all questions in the space provided. Write clearly and legibly, you will not get credit for illegible or incomprehensible answers. This is a closed book exam. However, each student is allowed to bring one page of notes to the exam. Print your name at the top of every page.

Question:	1	2	3	4	5	6	7	8	Total
Points:	12	21	12	8	15	12	10	10	100
Score:									

1. Multiple Choice Questions

(a) Which code would you use to instantiate a new TreeSet that could *only* hold Strings? (2)

A. `... = <String>TreeSet();`
B. `... = new TreeSet<String>;`
C. `... = TreeSet<String>();`
D. `... = new TreeSet<String>();`
E. `... = String<TreeSet>();`
F. `... = new String[TreeSet];`

(b) Which combination of modifiers *could not* be used together to modify a class? (2)

A. `public static abstract`
B. `protected abstract final`
C. `private static final`

(c) Which combination of modifiers *could* be used together to modify a member variable? (2)

A. `public static abstract`
B. `protected abstract final`
C. `private static final`

(d) What is the value of the following expression? `2 + 4 + "six"+ 8` (2)

A. `"6six8"`
B. `"24six8"`
C. `"six14"`
D. `20`
E. This expression would result in a compilation error.

(e) A member declared with a `protected` access modifier is *not always* visible to: (2)

A. the class in which it is declared.
B. classes in the same package as the class in which it is declared.
C. classes that extend the class in which it is declared.
D. parent classes of the class in which it is declared.
E. classes nested inside the class in which it is declared.

(f) Which of the following is true of an unchecked exception? (2)

A. It must be handled at compile time with a try/catch construct.
B. It is thrown because of unavoidable circumstances, such as a file not being found.
C. It extends `RuntimeException`.
D. It cannot be caught at runtime.

2. Why do the following code snippets not compile? (Explain in one sentence each.)

(a) (3)

```
int single = 1;
int double = 2;
```

(b) (3)

```
List<String> names = new List<String>();
```

(c) (3)

```
Set<double> values;
```

(d) (3)

```
Map<Integer, String> idToNameMap = HashMap<>();
```

(e) (3)

```
double pi = 3.14159
```

(f) (3)

```
if (x = 5) System.out.println(x);
```

(g) (3)

```
public static String myMethod(int x) {
    final String result = "Big";
    if(x < 10) {
        result = "Small";
    }
    return result;
}
```

3. Short answer

   (a) What is the keyword `instanceof` used for? (3)

   (b) When reading the API, how can you recognize a generic class? (3)

   (c) Name one method in the `Collections` class, and explain what it does. (3)

   (d) It is possible to declare a `main` method in every single class of a project. What would be the point of doing this? (3)

4. Consider the following classes.

```
public class Parent {
  public void method1(int i) {
  }
  public static void method2(int i) {
  }
  public void method3(int i) {
  }
  public static void method4(int i) {
  }
}
```

```
public class Child extends Parent {
  public void method1(float i) {
  }
  public static void method2(float i) {
  }
  public void method3(int i) {
  }
  public static void method4(int i) {
  }
}
```

   (a) Does `method1` in `Child` override, overload, or hide the method in `Parent`? (2)

   (a) ______________

   (b) Does `method2` in `Child` override, overload, or hide the method in `Parent`? (2)

   (b) ______________

   (c) Does `method3` in `Child` override, overload, or hide the method in `Parent`? (2)

   (c) ______________

   (d) Does `method4` in `Child` override, overload, or hide the method in `Parent`? (2)

   (d) ______________

5. Consider the following interface.

```
public interface TestInterface {
    void doStuff(String s);
    boolean isItTrue(int i, double x);
}
```

For each of the following:

- Does this class implement the interface?
- If it does not, what is wrong with the implementation?

(a) (3)

```
public interface TestImplementation implements TestInterface {

    public void doStuff(String s) {
    }

    public boolean isItTrue(int i, double x) {
        return true;
    }
}
```

(b) (3)

```
public class TestImplementation implements TestInterface {

    public void doStuff(String s) {
    }

    public boolean isItTrue(int i, double x) {
        return true;
    }

    public void doStuff(int i) {
    }
}
```

(c) (3)

```
public class TestImplementation extends TestInterface {

    public void doStuff(String s) {
    }

    public boolean isItTrue(int i, double x) {
        return true;
    }
}
```

(d) (3)

```
public class TestImplementation {

    public void doStuff(String s) {
    }

    public boolean isItTrue(int i, double x) {
        return false;
    }
}
```

(e) (3)

```
public class TestImplementation implements TestInterface {

    public boolean isItTrue(int a, double b) {
        return false;
    }

    public void doStuff(String x) {
    }

}
```

6. Consider the following classes. What is the output of this code? (12)

```
public class Foo {
    protected String a;
    protected int b;

    public Foo() {
        this("Goodbye");
    }

    public Foo(String a) {
        this(42, a);
    }

    public Foo(int a, String b) {
        this.a = b;
        this.b = a;
    }

    public void printStuff() {
        System.out.println(a);
        System.out.println(b);
    }
}
```

```
public class Bar extends Foo {
    protected double c;

    public Bar(int a, String b) {
        this(2.5, b, a);
    }

    public Bar(double a, String b, int c) {
        this.c = a*c;
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }

    public void printStuff() {
        super.printStuff();
        System.out.println(c);
    }

    public static void main(String[] args) {
        Foo x = new Bar(3, "Hello");
        x.printStuff();
    }
}
```

7. Consider the following class. What is the output of this code? (10)

```
public class Baz {
    private static int x = 2;
    private int y = 10;

    public void doStuff() {
        System.out.println(x);
        System.out.println(y);
        x += y;
        y--;
    }

    public static void main(String[] args) {
        Baz b1 = new Baz();
        b1.doStuff();
        b1.doStuff();

        Baz b2 = new Baz();
        b2.doStuff();
        b1.doStuff();
        b2.doStuff();
    }
}
```

8. Write a method that takes a `Collection` of `String` objects (any type of collection, not a specific implementation) and returns the average number of characters in all the `String`s. In other words, return the average of the lengths of the `String`s in the `Collection`. If the collection is empty, return -1. (10)