Name:_____ NetID:_____

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

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	14	10	16	6	10	12	12	10	10	100
Score:										

1. Indicate if the following statements are true or false.
   (a) A class can implement any number of interfaces. (2)
   (a) _____
   (b) A class can extend any number of classes. (2)
   (b) _____
   (c) An abstract class cannot contain any method implementations. (2)
   (c) _____
   (d) It is possible to use an interface as a type. (2)
   (d) _____
   (e) Any reference type can be assigned to an `Object` variable. (2)
   (e) _____
   (f) An unchecked exception cannot be caught with a try/catch construct. (2)
   (f) _____
   (g) Every class has a default no-argument constructor. (2)
   (g) _____

2. Consider the following classes. For each specified method, does the method in `Child` override, overload, or hide the method in `Parent`?

```
public class Parent {
  public void foo1(int i) { }
  protected static void foo2(int i) { }
  protected void foo3(int i) { }
  protected static void foo4(int i) { }
  public void foo5(int i) { }
}
```

```
public class Child extends Parent {
  public void foo1(float i) { }
  public static void foo2(float i) { }
  public void foo3(int i) { }
  protected static void foo4(int i) { }
  public void foo5(int i) { }
}
```

   (a) `foo1` (2)
   (a) _____
   (b) `foo2` (2)
   (b) _____
   (c) `foo3` (2)
   (c) _____
   (d) `foo4` (2)
   (d) _____
   (e) `foo5` (2)
   (e) _____

3. For the following questions, select the single *best* answer.

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

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

(b) Which combination of modifiers *could* be used together to modify a variable inside a class? (2)

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

(c) Which methods can access the private members of a class? (2)

A. Only static methods of the same class.
B. Only final methods of the same class.
C. Only methods defined in the same class.
D. Only methods defined in the same package (including those in the class itself).
E. Only methods within the same class or its children.
F. Only methods within the same class, its children, or in the same package.

(d) Which code would you use to construct a new ArrayList that could *only* hold Double objects? (2)

A. `new Double[];`
B. `new Double[ArrayList];`
C. `new ArrayList<Double>();`
D. `new ArrayList();`
E. `new ArrayList<Double>;`
F. `<Double>ArrayList();`
G. `Double<ArrayList>();`
H. `ArrayList<Double>();`

(e) What is the value of the following expression? (2)

```
1 + "2" + 3 * 4 + 5
```

A. `15`
B. `20`
C. `"20"`
D. `"1217"`
E. `"12125"`
F. `"12345"`
G. The value of this expression is undefined.
H. This expression would result in a compilation error.

(f) Which interface would be the best choice to hold a collection of unique dates? (2)

A. `Collection`
B. `Deque`
C. `List`
D. `Map`
E. `Set`
F. `Queue`

(g) Which type could `foo` be in the following code snippet? (2)

```
Object obj = foo.get(0);
```

A. `Collection`
B. `List`
C. `Set`
D. `Deque`
E. `SortedSet`
F. `Queue`

(h) Which type would be the best choice if you wanted to associate ISBNs with prices for the corresponding books? (2)

A. `Collection`
B. `Deque`
C. `List`
D. `Map`
E. `Set`
F. `Queue`

4. Why don't the following code snippets compile? Select the single correct answer for each.

(a) (2)

```
public class MyClass {
    private static final int x = 10;

    public static void main(String[] args) {
        x++;
        System.out.println(x);
    }
}
```

A. Cannot access private variable x from a public method.
B. Cannot access x without an instance of MyClass.
C. Variable x is out of scope in the main method.
D. Variable x is a constant, so cannot be incremented in main.
E. The println method expects a String, not an int.
F. Some other error.
G. This code will successfully compile.

(b) (2)

```
public static boolean foo(short n) {
  int default = 10;
  return n < default;
}
```

A. Cannot compare int and short variables.
B. Cannot use a keyword as a variable name.
C. Invalid parameter type.
D. Return value does not match method return type.
E. Did not initialize value of n.
F. Some other error.
G. This code will successfully compile.

(c) (2)

```
public static void bar(int n) {
  if(n < 5) {
    System.out.println("Small");
  } else {
    throw new Exception("Big!");
  }
}
```

A. No value returned in method.
B. Invalid test in if statement.
C. Did not initialize value of n.
D. Did not handle checked exception.
E. Did not declare method throws an unchecked exception.
F. Some other error.
G. This code will successfully compile.

5. Why don't the following code snippets compile? Select the single correct answer for each.

(a) (2)

```
Collection<Double> = TreeSet<>();
```

A. Cannot use Double as generic type parameter.
B. Cannot assign a TreeSet to a Collection variable.
C. Cannot use interface Collection as variable type.
D. Missing type parameter on right hand side of assignment.
E. Some other error.
F. This code will successfully compile.

(b) (2)

```
List<boolean> truthValues;
```

A. The list is not initialized.
B. Cannot use primitive type boolean as generic type parameter.
C. Cannot use interface List as variable type.
D. List is an interface and cannot be instantiated.
E. Some other error.
F. This code will successfully compile.

(c) (2)

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

A. Cannot use Integer as generic type parameter.
B. Cannot use String as generic type parameter.
C. Cannot use interface Map as variable type.
D. Missing type parameter on right hand side of assignment.
E. Some other error.
F. This code will successfully compile.

(d) (2)

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

A. Cannot use String as generic type parameter.
B. Cannot use interface Set as variable type.
C. Set is an interface and cannot be instantiated.
D. Missing type parameter on right hand side of assignment.
E. Some other error.
F. This code will successfully compile.

(e) (2)

```
List<String> bookTitles
```

A. The list is not initialized.
B. Cannot use String as generic type parameter.
C. Cannot use interface List as variable type.
D. List is an interface and cannot be instantiated.
E. Some other error.
F. This code will successfully compile.

6. Consider the following interface.

```
public interface Nonsense {
    void returnNothing(String s, int n);
    boolean giveBoolean(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 class Junk {
    public void returnNothing(String s, int n) { }
    public boolean giveBoolean(double x) { return false; }
}
```

(b) (3)

```
public class Junk implements Nonsense {
    public boolean giveBoolean(double d) { return false; }
    public void returnNothing(String x, int y) { }
}
```

(c) (3)

```
public class Junk implements Nonsense {
    public void returnNothing(int n, String s) { }
    public boolean giveBoolean(double x) { return true; }
}
```

(d) (3)

```
public class Junk implements Nonsense {
    public void returnNothing(String s, int n) { }
    public boolean giveBoolean(double x) { return true; }
    public void returnNothing(int i) { }
}
```

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

```
public class Foo {
    protected int a = 10;
    protected String b = "hello";

    public Foo() {
        this(251, "CS");
    }

    public Foo(String a) {
        this(2017, a);
    }

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

    public void doStuff(String c) {
        b = c;
        doStuff();
    }

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

```
public class Bar extends Foo {
    protected double c = 2.5;

    public Bar(String c) {
        this.c = a - 0.5;
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }

    public void doStuff() {
        a++;
        super.doStuff();
        System.out.println(c);
    }

    public static void main(String[] args) {
{       Foo x = new Bar("Midterm");
        x.doStuff("Exam");
    }
}
```

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

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

    public Baz(int x) {
        this.x = x;
    }

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

    public static void main(String[] args) {
        Baz b1 = new Baz(20);
        b1.doStuff();

        Baz b2 = new Baz(15);
        b2.doStuff();

        b1.doStuff();
        b2.doStuff();
        b1.doStuff();
    }
}
```

9. Write a method that takes a `Collection` of `String` objects (this must accept any type of collection, not just a specific implementation) and returns the longest of the `String`s beginning with the character `'M'` (uppercase only). Any `String` that does not begin with `'M'` should be ignored. If the collection does not contain any `String` objects beginning with the character `'M'`, return `null`. (10)