University of Maryland College Park
Dept of Computer Science
CMSC 132
Exam #1

First Name (PRINT): _____

Last Name (PRINT): _____

University ID: _____

Section: _____

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Your signature: _____

Instructions

- *This exam is a closed-book and closed-notes exam.*
- *Total point value is 100 points*
- *The exam is a 50 minute exam.*
- ***WRITE NEATLY**. If we cannot understand your answer, we will not grade it (i.e., 0 credit).*

Grader Use Only

Page 2	(24)	
Page 3	(13)	
Page 4	(16)	
Page 5	(11)	
Page 6-7	(15)	
Page 8-10	(21)	
Total	(100)	

1. (2 points) What is the **name** of the method that must be included in any class implementing the `Comparator` interface?

2. (2 points) When does a (non-static) initialization block run?

3. (2 points) When does a static initialization block run?

4. (2 points) What keyword can be placed before a class declaration to prevent anyone from extending the class?

5. (2 points) What keyword can be placed at the beginning of a method prototype to prevent anyone from overriding the method?

6. (2 points) If you have 100% flow coverage, do you necessarily have 100% conditional coverage?

   Circle one: **YES** **NO**

7. (2 points) If you have 100% conditional coverage, do you necessarily have 100% flow coverage?

   Circle one: **YES** **NO**

8. (2 points) Can code coverage results demonstrate that a set of unit tests is incomplete?

   Circle one: **YES** **NO**

9. (2 points) Can code coverage results demonstrate that a set of unit tests is complete?

   Circle one: **YES** **NO**

10. (2 points) Which kind of exception is thrown in a case where the programmer has made a serious error that should be corrected?

    Circle one: **checked** **unchecked**

11. (2 points) Which kind of exception is thrown in a case where something unusual has happened, but it was not the result of buggy code?

    Circle one: **checked** **unchecked**

12. (2 points) What is the **name** of the method that is used in Java as a *destructor*?

13. (2 points) An abstract class has an advantage over an interface in that it

    facilitates_____. (Fill in the blank with one word.)

14. (2 points) Java allows multiple inheritance.

    Circle one: **TRUE** **FALSE**

15. (2 points) What kind of *binding* does Java use?

16. (2 points)

```
String x = "pencil";
x = "pen";
```

    After the code fragment above has run, what is the technical term for the String that says "pencil"?

17. (2 points) What is the visibility specifier used for a member that should be visible within the current package and also visible in any subclass of the current class?

18. (3 points) Assume that the class `Spaceship` contains an inner class called `Robot`, and that each of these classes has a constructor that takes no parameters.

    a. Write a statement that instantiates a `Spaceship` and assigns it to a variable.

    b. Write a statement that instantiates a `Robot` that is associated with the `Spaceship` from the previous question. The code you write must compile and run when located *outside* the `Spaceship` class.

For the next few questions, assume there is a class called `Lifeform`, that the class `Fish` extends `Lifeform`, and that the (unrelated) class `Plant` also extends `Lifeform`.

19. (12 points) Assume the following statements have run:

```
Fish fish = new Fish();
Lifeform life = new Lifeform();
Lifeform x = new Fish();
```

For each statement below, circle one of the outcomes listed. (Assume each statement below is run independently of the others.)

a. `Plant a = x;`

Won't Compile **Throws Exception** **Works Fine**

b. `Plant b = (Plant) x;`

Won't Compile **Throws Exception** **Works Fine**

c. `Lifeform c = fish;`

Won't Compile **Throws Exception** **Works Fine**

d. `Fish d = (Fish) life;`

Won't Compile **Throws Exception** **Works Fine**

e. `Fish e = (Fish) x;`

Won't Compile **Throws Exception** **Works Fine**

f. `Plant p = (Plant) fish;`

Won't Compile **Throws Exception** **Works Fine**

20. (2 points) Assume there is a method called `move` in the `Lifeform` class that has been overridden in the `Plant` class. Consider the code fragment below:

```
Lifeform x = new Plant();
x.move();
```

Which version of the `move` method will run? (Circle one.)

`Lifeform version` **`Plant version`**

21. (2 points) Assume you have written a method with the prototype below:

```
public void foo(Lifeform x) {…}
```

Which of the following classes of objects could be passed in as an argument for this method? (Circle all that apply.)

`Fish` **`Plant`** **`Lifeform`** **`Object`**

22. (6 points) Assume that the variable `list` is of type `ArrayList<Lifeform>` and that it has been populated with various lifeforms. Assume that the `Fish` class has an instance method called `swim`. Write a for-each loop that will call the `swim` method on every `Fish` in the list.

23. (5 points) Consider the interface below:

```
public interface Friendly {
   public void sayHello();
}
```

Declare a variable of type `Friendly`, and use an anonymous inner class to assign an object to your variable. Your object must implement the `sayHello` method by printing the word "Hi" on the console.

24. (15 points) You must use an inner class for this question, but you may **not** use an *anonymous* inner class. We are writing a class called `StringDisector`, which wraps a `String` and has the ability to iterate over the characters in the String, as illustrated by the following example:

Example Code	**Output**

`StringDisector sd = new StringDisector("CMSC");`
`for (Character c : sd) {`
`  System.out.println(c);`
`}`
	`C`
	`M`
	`S`
	`C`

Complete the class below so that it will work as illustrated in the example above. **(You do not have to implement the remove method for the Iterator.)**

Hint: Recall the `charAt` method of the `String` class, which accepts an integer as an argument and returns the character in the String whose index is equal to that integer.

```
public class StringDisector implements Iterable<Character>{
      private String s;
      public StringDisector(String s) {
            this.s = s;
      }
```

[You may continue your answer on this page.]

25. (21 points) Consider the following two classes, which are only partially shown:

```
public class Vehicle implements Cloneable {

      private String description;

      public Vehicle(String description) {
            this.description = description;
      }
      .
      .
      .
}
```

```
public class Engine implements Cloneable{

      private int horsepower;

      public void tune() {
            horsepower++;
      }
      .
      .
      .
}
```

You may assume that each of the classes above will behave appropriately when either clone or equals is called on them.

On the following pages, write a class called `Car`, which extends `Vehicle` and implements the `Cloneable` interface. Include the following features (and no others):

- A private instance variable, `engine`, of type Engine.
- A constructor that accepts two parameters: An Engine, and a String. These parameters will be used to initialize the Car's `engine` and `description` fields, respectively.
- A carefully implemented `equals` method. In order to be considered equal, two Cars must have equal Engines and equal descriptions.
- A carefully implemented `clone` method.

[You may continue your answer on this page.]