Written Test

CSE 1030 3.0

Section A, Summer 2011

Family Name: _____

Given Name(s): _____

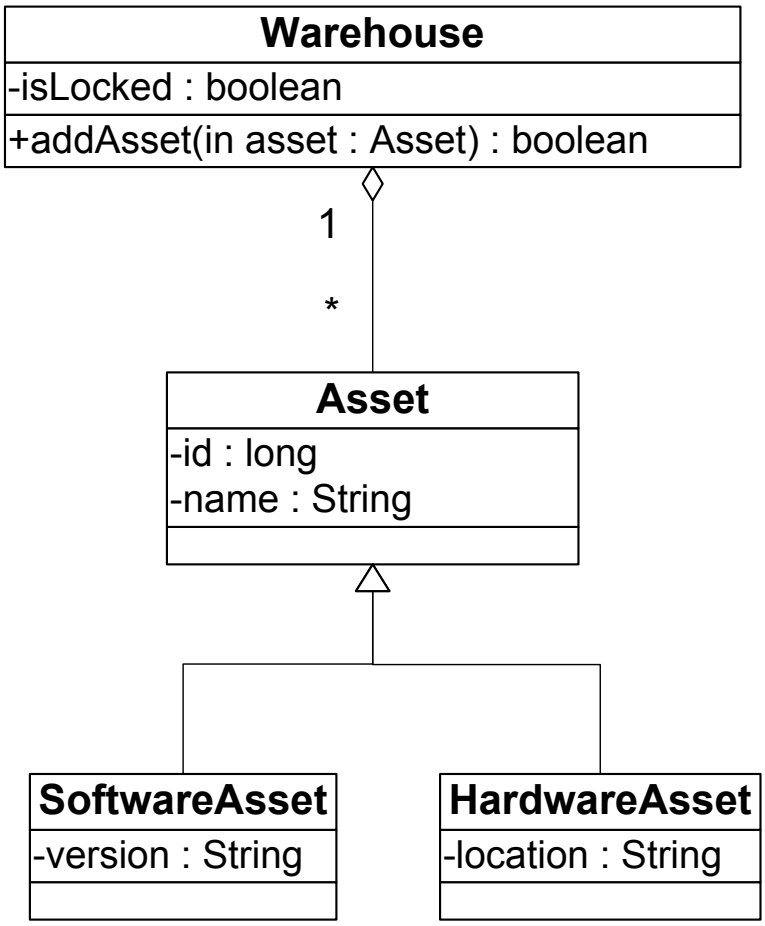
Student Number: |___ |___ |___ |___ |___ |___ |___ |___ |___ |

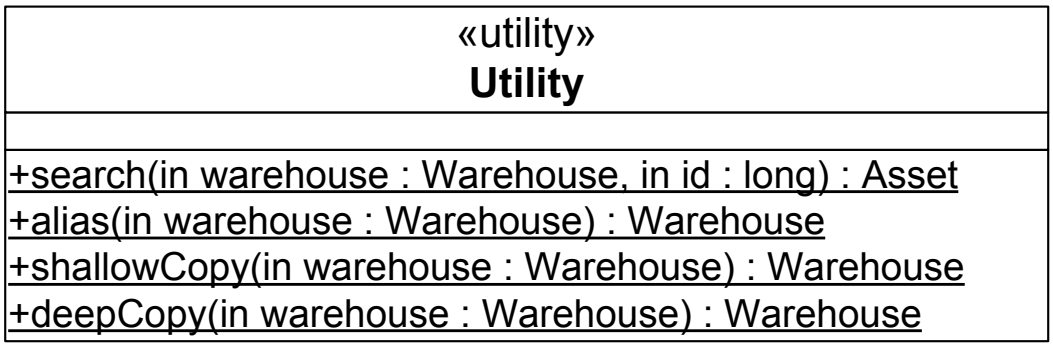
Guidelines and Instructions:

1. This is a 80-minute test. You can use the textbook, but no electronic aids such as calculators, cellphones etc.
2. Answer questions in the space provided. If you need more space, use the back of the page. Clearly indicate that your answer continues on the back of the page.
3. Write legibly. Unreadable answers will not be marked.
4. Leave your ID on the desk. A sign-up sheet will be distributed during the test. By signing it, you acknowledge that you are registered in the course and you are the owner of the associated ID.
5. Keep your eyes on your own work. At the discretion of the invigilators, students may be asked to move.

Question	Out of	Mark
Q1	15	
Q2	20	
Q3	15	
Q4	20	
Total	70	
Letter grade		

All questions in this exam refer to the following set of classes. Study them carefully before answering any questions.





Some of the attribute and methods in these classes are self-explanatory. This page provides extra information on the remaining attributes, as well as the `search`, `alias`, `shallowCopy` and `deepCopy` methods.

The `Warehouse` class stores information about company assets. The `isLocked` attribute indicates whether the warehouse is locked, which means that it is impossible to add assets.

The company has two types of specialized assets:

- Software
- Hardware

Every asset has a `name` and an automatically generated unique `id` number.

Software assets have the attribute `version` which contains information about the software version, such as `"Microsoft Word 2008"`.

Hardware assets have the attribute `location` which contains information about the location of the asset, such as `"Server Room"`.

The `search` method in class `Utility` finds and returns an asset that matches a given `id`. If no asset matches the given `id`, `null` is returned.

The `alias`, `shallowCopy` and `deepCopy` methods in class `Utility` create a copy of the `Warehouse` object in a manner indicated by their name.

Instructions pertaining to all questions in this exam

- Minor syntax errors and outputting formatting problems in your answers **will not** affect your mark. Concentrate on concepts rather than syntax.
- During the exam you may need to introduce or modify various methods. To do so, you should write:

  `-- ClassName:Method`

  followed by the method declaration. For instance, in order to add a `remove` method to class `Warehouse` you should write:

```
-- Warehouse:remove

public boolean remove(Asset asset) {
  return this.getAssets().remove(asset);
}
```

Q1. [15 marks] For this question, you must implement the constructors of classes `Asset` and `SoftwareAsset`. You must also show other parts of these two classes that are necessary to ensure that each asset has a unique `id`.

```
public class Asset
{
  String name;
  long id;
  static long nextID = 0;

  public Asset(String name) {
    this.setName(name);
    id = Asset.nextID++;
  }

    public  void setName(String name) {
    this.name = name;
  }

    private void setID(long id) {
    this.id  = id;
    }
}
```

```
public class SoftwareAsset extends Asset
{
  String version;
  public SoftwareAsset(String version, String name)
  {
    super(name);
    this.setVersion(version);
  }
  public void setVersion(String version) {
    this.version= version;
  }

}
```

The setId must be a private method so that uniqueness of asset id cannot be compromised by a client

5 marks inheritance implemented
5 marks constructors implemented
5 marks uniqueness

Q2. [20 marks] Implement the three copying methods of the `Utility` class, as shown by the method signatures in sections (a), (b), and (c).

You are free to modify any of the given classes, or re-use them as is.

(a) [3 marks]

```
public static Warehouse alias (Warehouse w) {

  return w;
}
```

(b) [7 marks]

```
public static Warehouse shallowCopy (Warehouse w) {


  Warehouse copy = new Warehouse();
  copy.setLock(w.getLock());
  copy.setAssets(w.getAssets());
  return copy;
}
```

5 marks copy attribute
2 marks create new instance

(c) [10 marks]

```
public static Warehouse deepCopy (Warehouse w) {

  Warehouse copy = new Warehouse();
  copy.setLock(w.getLock());
  for (Asset asset:w.getAssets())
  {
      Asset copyAsset;
      if (asset instanceof HardwareAsset) {
              copyAsset = new HardwareAsset((HardwareAsset)asset);
      } else if (asset instanceof SoftwareAsset) {
              copyAsset = new SoftwareAsset((SoftwareAsset)asset);
      } else {
              copyAsset = new Asset(asset);
      }
      copy.addAsset(copyAsset);
  }
  return copy;
}
-- Asset:Asset (add copy constructor)

public Asset(Asset asset) {
  this.setName(asset.getName());
  this.setID(asset.id);
}
-- HardwareAsset:HardwareAsset (add copy constructor)

public HardwareAsset(HardwareAsset asset) {
  super(asset);
  this.setLocation(location);
}
-- SoftwareAsset:SoftwareAsset (add copy constructor)

public SoftwareAsset(SoftwareAsset asset) {
  super(asset);
  this.setVersion(location);
}
```

5 marks copy constructor
5 marks copy assets

Q3. [15 marks] Implement the `search` method of the `Utility` class without using any loops.

```
public static Asset search (Warehouse w, long id) {
  if (w.getAssets().isEmpty())
    return null;

 if (w.getAssets().get(0).getID() == id)
    return w.getAssets().get(0);

  Warehouse copy = Utility.deepCopy(w);
  copy.getAssets().remove(0);
  return search(copy,id);
}
```

The deep copy is required because the recursion modifies the warehouse collection

5 marks base cases
5 marks divide-and-conquer idea demonstrated
5 marks recursion idea demonstrated

Q4. [20 marks] What is the output of the following app. **Justify** your answer.

```
public static Warehouse method (int a, int b, Warehouse w1, Warehouse w2)
{
  a = b;
  b = 5;
  w1.setLock(w2.getLock());
  w1 = w2;
  w2.setLock(true);
  return w1;
}

public static void main(String[] args)
{
  Warehouse w1 = new Warehouse();
  Warehouse w2 = new Warehouse();
  w1.setLock(false);
  w2.setLock(false);
  int a = 0;
  int b = 0;
  Warehouse w3 =  method (a, b, w1, w2);
  System.out.println("a= " + a);
  System.out.println("b= " + b);
  System.out.println("w1.getLock= " + w1.getLock());
  System.out.println("w2.getLock= " + w2.getLock());
  System.out.println("w3.getLock= " + w3.getLock());
}
```

a=0

b=0

w1.getLock= false

w2.getLock= true

w3.getLock = true

The change of primitive arguments did not effect on caller due to pass-by-value mechanism

State of w1 change did not change since both w2 and w1 has the same value of isLock attribute

The change of w1 reference did not effected on caller due to pass-by-value mechanism

State of w2 changed due to w2.setLock(true), so output of w2.getLock is true w3 reference on w2 object due to method returns w1 which references to w2 object

5 marks correct output
5 marks correct explanation passing parameters
5 marks correct explanation return value
5 marks correct explanation references manipulation