[redacted]

- Turn off your cell phone to avoid losing a letter grade when it rings.
- Use the amount of space provided to gauge how much you should write. Brevity is the soul of points. Points are not related to wit..
- Legibility counts, so be neat. If your writing is smaller than the typeface of this exam, I may deduct points.
- Points may be deducted for irrelevant, meaningless, or contradictory statements (and of course, just plain false statements). Please be sure to answer the question I asked!
- Do not complicate an example. Do not make up features of an example unless directed to do so. Simple is best!
- Some questions look hard at first, but if you for(i=0;i<3;i++) breatheDeeply(); you realize it is simpler than you first thought.
- **Do not change code I have written** unless explicitly directed to do so.

1 12pts waterfall $ agile
2 20/20pts Basic interface mode UML Diagram (check over) 20/20
3 6/8 # birds.
4 36/36 Check duplicates in arraylist collection / Hashset Collection
56
5 12/12 [illegible] object from Hashset
6 12pts git.

1. (12 pts) Write two major conceptual differences between the Agile and Waterfall approaches, as discussed in class.(Hint: conceptual differences are **not** about concrete differences such as "one has a sprint".)

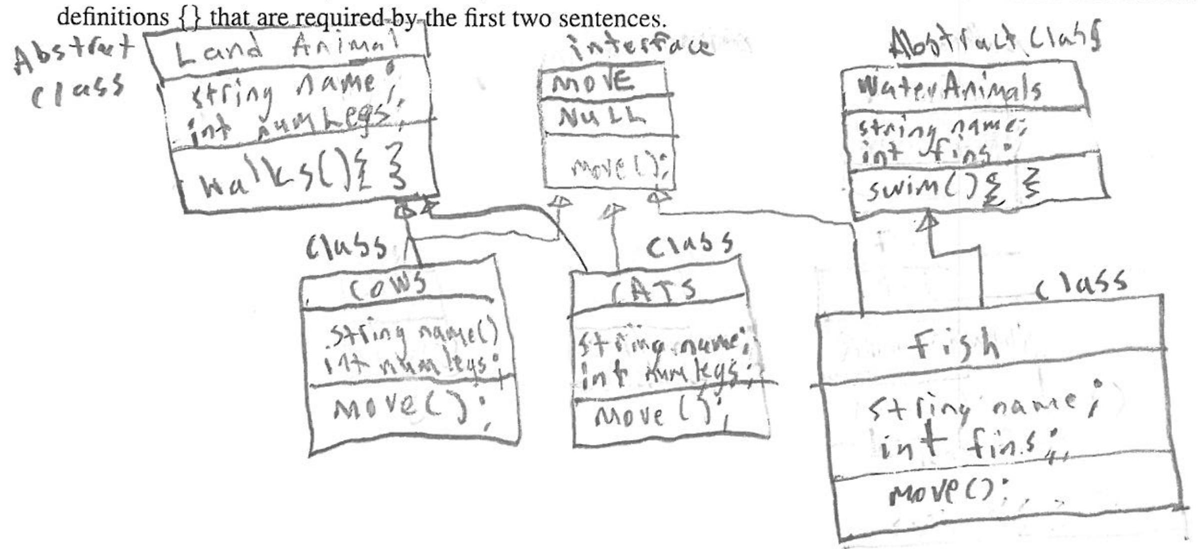
(1) Waterfall has minimal changes to the design but agile you can change essentially everything up, including requirements to reach client' desires.

(2) Alot of initial planning in waterfall towards one main plan, and alot of continuos and changing & adapting planning in agile.

→ Waterfall cycles 1 by 1 Analysis, planning, coding, Verify sequentially, whil agile cycles continuosly Parrelle

2. (20 pts) To move, cows and cats walk, while fish swim. However, all three move.

Represent the preceding two sentences in a UML-style class diagram that has five or six nodes. Be sure to denote whether a node is a class, abstract class, or interface. Be sure to show locations of the method declarations and definitions {} that are required by the first two sentences.



3. (8 pts) Name the four birds from our client's presentation, and state whether they are migratory or not.

Blue Hen : Migratory
Red Bird : migratory
Orayuwa Bird : not migratory
Blue Bird : Migratory.

4. (36 pts) Create a class Dog, and then make a Collection of Dogs such that no duplicate dogs will be in the collection. Dogs are considered duplicates if they have the same name. Assume all imports are provided.

```
public class Dog {

        String name;

        //4 pts Write a constructor


        //4 pts When Dogs are printed, we see their name.

            NSS so IDK if, do we use Hashset, and override equals, &

        //16 pts Dogs are considered duplicates if they have the same name.

                Class Dog{ this.name = name;}
                Public String getname(){ return name;}
                @override
                Public boolean equals( object o){
                      if (o instanceof Dog){
                         other Dog = (Dog) o;
                         return other.getname(). ToString(this.getname());
                      }else{
                         return false;
                      }
                @override
                 Public String Hashcode{
                      return name
                 }

it.hasNext.getName

        //8 pts Create a Collection, and write code that will demonstrate when it run
        // that your collection does not add duplicate Dogs.
        public static void main(String[] args) {

                Hashset<Dogs> coll = new Hashset<>();
                coll.add(new Dog("Buddy"));
                coll.add(new Dog("Buddy"));
                coll.add(new Dog("Rex"));
                if(coll.size() == 2){
                   System.out.println("Collection doesn't duplicate");
                }else{
                   System.out.println("Collection duplicates");

                //4 pts Print the Collection (use minimal code).
                for (Dogs d: coll){
                   System.out.println(d.getname());
                }

        }
}
```

5. (12 pts) You are given a HashSet of Birds named **flock**. All birds have a boolean method **migrates**() that returns true if and only if the bird is a migratory bird. Using this method (do not write the method), write a few lines of Java code (not a method) to remove all non-migratory birds from **flock**.

~~Hashset <Birds> flock = new HashSet<>();~~

```
Iterator <Birds> it = flock.iterator();
while(it.hasnext()){
   if (it.migrates() == false){
          flock.rem(it);
   }
}
```

6. (12 pts) Assume you have a single file, a.txt, in a directory. Initialize a repo and show the commands required to get the following tree (commit messages have been removed; be sure to match asterisk count as well as branch shape). Note end state of the tree!

```
* commit 14a00be7934bcd03c38542e645f3e8a57bea4652 (master)
| Date:   Wed Apr 10 13:49 2019 -0400
|
|
| * commit cf84d43f26db4e8ea1ee89d91e23e3df186f80d1 (HEAD -> feature)
|/  Date:   Wed Apr 10 13:49 2019 -0400
|
|
* commit 043223c0996ffc6e6c265794c1b5b6a1a9297220
  Date:   Wed Apr 10 13:49 2019 -0400
```

```
git init
touch a.txt
cat >> a.txt → "first line"
git add -A
git commit -m "first commit"
git branch feature
git checkout feature
cat >> a.txt → "second line"
git add -A
git commit -m "second commit"
git checkout master
cat >> a.txt → "third line"
git add -A
git commit -m "Final, third commit"
```