Most difficult part of assignment:
Getting comfortable with Java syntax for Object Oriented Programming

Status:
Completed

Lines of Code:
```
public class Thing
{
   private String name;
   public Thing(String name)
   {
      this.name = name;
   }
   public String toString()
   {
      if (getClass().getSimpleName().equals("Thing"))
         return name;
      else
         return name+" "+getClass().getSimpleName();
   }
}
public abstract class Creature extends Thing
{
   private String name;
   private Thing aThing;
   public Creature(String name)
   {
      super(name);
   }
   public void eat(Thing aThing)
   {
      System.out.println(super.toString()+" has just eaten a "+aThing+".");
   }
   public abstract void move();
   public void whatDidYouEat()
   {
      if (aThing != null)
         System.out.println(super.toString()+" has eaten a "+aThing);
      else
         System.out.println(super.toString()+" has had nothing to eat!");
   }
}
```

```
public class Tiger extends Creature
{
   private String name;
   public Tiger(String name)
   {
      super(name);
   }
   public void move()
   {
      System.out.println(super.toString()+" has just pounced.");
   }
}
public class Ant extends Creature
{
   private String name;
   public Ant(String name)
   {
      super(name);
   }
   public void move()
   {
      System.out.println(super.toString()+" is crawling around.");
   }
}
public interface Flyer
{
   void fly();
}
public class Fly extends Creature implements Flyer
{
   private String name;
   public Fly(String name)
   {
      super(name);
   }
   public void eat(Thing aThing)
   {
       if (aThing instanceof Creature)
         System.out.println(super.toString()+" won't eat a "+aThing+".");
      else if (aThing instanceof Thing)
         super.eat(aThing);
   }
   public void move()
   {
      fly();
```

```
    }
    public void fly()
    {
        System.out.println(super.toString()+" is buzzing around in flight.");
    }
}
public class Bat extends Creature implements Flyer
{
    private String name;
    public Bat(String name)
    {
        super(name);
    }
    public void eat(Thing aThing)
    {
        if (aThing instanceof Thing)
            System.out.println(super.toString()+" won't eat a "+aThing+".");
        else if (aThing instanceof Creature)
            super.eat(aThing);
    }
    public void move()
    {
        fly();
    }
    public void fly()
    {
        System.out.println(super.toString()+" is swopping through the dark.");
    }
}
public class TestCreature
{
    public static final int CREATURE_COUNT=6;
    public static final int THING_COUNT=10;
    public TestCreature()
    {}
    public static void main(java.lang.String[] args)
    {

        Thing[] t = new Thing[THING_COUNT];
        for ( int l=0; l<THING_COUNT-4; l++)
        {
            t[l] = new Thing("newthing"+l);
        }
        for( int m=6; m<THING_COUNT; m++)
        {
          t[m] = new Tiger("newtigthing"+m);
        }
        System.out.println("Things:\n");
        for (int i=0; i<THING_COUNT; i++)
        {
          System.out.println(t[i]);
        }
        System.out.println();
        Tiger[] tig = new Tiger[CREATURE_COUNT];
        for ( int k=0; k<CREATURE_COUNT; k++)
        {
          tig[k] = new Tiger("NewTiger"+k);
        }
        System.out.println("Tigers:\n");
        for (int j=0; j<CREATURE_COUNT; j++)
        {
          System.out.println(tig[j]);
        }
        System.out.println();
        Thing[] th = {new Bat("Bunty"), new Thing("Locomotive"), new Ant("Fruit"), new
Thing("Banana")};
        System.out.println("Things:\n");
        for (int i=0; i<th.length; i++)
        {
          System.out.println(th[i]);
        }
        System.out.println();
        Creature[] c = {new Ant("Andy the"), new Bat("Barry the"), new Tiger("Tigga the"), new
Fly("Flint the")};
        System.out.println("Creatures:\n");
        for (int p=0; p<c.length; p++)
        {
          c[p].move();
          c[p].eat(th[p]);
        }
        System.out.println();
    }
}
```