Interfaces and Polymorphic Fundamentals

1. An interface is typically a list of methods required for any class that chooses to implement an interface. Although Java copied a lot of characteristics from C++, Interfaces is one concept that Java added (i.e. C++ really doesn't have the concept of an interface).
2. Consider the following hypothetical interface:

```
package life;


public interface Living {
            public void move();
            public void eat();
            public void born();
            public void die();
}
```

3. A class can implement an interface like the Living interface above with a statement like:

```
Tree implements Living
{ ....
}
```

4. When a class implements an interface, it is required to have all of the methods in the interface (i.e. move, eat, born, die, etc.)
5. Go ahead and create classes like Tree, Snake and Cow that all implement the Living interface.
6. With these classes in place, it is now possible to create an array of Living things. For example, the following code could be possible:

```
Living[] livingArr = new Living[5];

livingArr[0] = new Tree();
livingArr[1] = new Snake();
// .....
```

7. Try it out, and go through the livingArr and call the methods for all of the objects which implement the Living interface (i.e. move, eat, born, die).

Sample Program using Living Interface

```
// Each of the following needs to be in a separate file
//*********************************************************

package life;

public interface Living {
    public void move();
    public void eat();
    public void born();
    public void die();
}

//*********************************************************

package life;

public class Snake implements Living{

        public void move() {
            System.out.println("I like to slither on the ground");
        }
```

```
        public void eat() {
            System.out.println("I like to just open wide and suck my prey in whole ... yumm");
        }

        public void born() {
            System.out.println("I once was hatched as an egg and had to break my way out. ");
        }

        public void die() {
            System.out.println("People who are afraid of me sometimes smash me with shovels ");
        }

    }

//*********************************************************

package life;

public class Cow implements Living{

        public void move() {
            System.out.println("I walk on all fours");
        }

        public void eat() {
            System.out.println("I like chew my cud");
        }

        public void born() {
            System.out.println("I once was born as a calf ");
        }

        public void die() {
            System.out.println("People who like hamburger usually do me in. ");
        }

    }



//*********************************************************

package life;

public class Tree implements Living{

    public void move() {
        System.out.println("I really don't move, but I do sway");
    }

    public void eat() {
        System.out.println("I suck nutrition from my soil and take in light from the sun");
    }

    public void born() {
        System.out.println("I was born from a seed that dropped one day");
    }

    public void die() {
        System.out.println("Sometimes you cut me down, sometimes I can live a long time. ");
    }

}


//*********************************************************
```

```
package life;

public class GameOfLife {

    public static void main(String[] args) {
        Living[] livingArr = new Living[5];

        livingArr[0] = new Tree();
        livingArr[1] = new Cow();
        livingArr[2] = new Snake();
        livingArr[3] = new Tree();
        livingArr[4] = new Snake();

        for (int i=0; i < livingArr.length; i++)
        {
            System.out.println("Item: "+i);
            livingArr[i].born();
            livingArr[i].eat();
            livingArr[i].move();
            livingArr[i].die();

            System.out.println("********");
        }

    }

}
```

Polymorphism:

In cps161, we had several Polymorphism examples:

1. **Zoo: which contained and Array of Animal ... Animal was abstract, with Snake, Cow, and Horse extending Animal**

2. **Shape abstract class with Circle, Rectangle, etc. extending Shape**

3. **Inheritance/polymorphism Chess Game:**

   ChessPiece class with Rook, Knight, etc extending ChessPiece

You may want to access the a cps161 web site to review. One hint, you can often take the URL to your cps261 class and change the "261" to "161" and this will normally give you a cps161 web site during the same semester as your cps261 class.

Zoo: Animal, Snake, Cow, Horse

```
public abstract class Animal
{
    private String name;
    private double weight;
    private int age;

    Animal()
    {
        this("No Name", 500., 1);
    }
```

```
    Animal(String n, double weight, int age)
    {
        name = n;
        this.weight = weight;
        this.age = age;
    }
    double getWeight()
    {
        return weight;
    }

    abstract String makeNoise();

    public String toString()
    {
        return name + " weight="+weight +" age="+age;
    }
}

class Cow extends Animal
{
    private int num_spots=1;
    Cow()
    {

    }
    Cow(String name, double weight, int age, int num_spots)
    {
        super(name, weight, age);
        this.num_spots = num_spots;
    }
    String makeNoise()
    {
        return "Mooo";
    }
    public String toString()
    {
        return super.toString() + " num spots="+num_spots;
    }

}
class Horse extends Animal
{
    ....

}


class Snake extends Animal
{
    ....
}

public class Zoo
{
    private int actual_num_animals=0;
    private int num_cages;
    private Animal[] animals;

    Zoo()
    {
        this(20);
    }
    Zoo(int num_cages)
    {
        ...
    }
```

```
    void add(Animal a)
    {
       ....
    }

    double total_weight()
    {
       ....
    }

    void make_all_noises()
    {
        for (int i=0; i < actual_num_animals; i++)
            System.out.println( animals[i].makeNoise());
    }
    void print_all_animals()
    {
       ....

    }
     public static void main(String[] args)
        {
            Zoo z = new Zoo();
            Snake sly = new Snake("Sly", 5.0 , 2, 2);
            Snake sly2 = new Snake("Slyme", 10.0 , 1, 2);
            Cow blossy = new Cow("Blossy", 900., 5,  10);
            Horse prince = new Horse("Prince", 1000., 5, 23.2);

            // Following not allowed because Animal is abstract
            //Animal spot = new Animal("Spot", 10., 4);

            z.add(sly);
            z.add(sly2);
            z.add(blossy);
            z.add(prince);

            z.make_all_noises();
            ....

        }
}
```

Last Updated: January 7, 2014 10:37 PM