CECS 277 – Lecture 4 – Interfaces

Interfaces – An interface specifies a behavior or ability for a class. They are similar to an abstract class that contains all abstract methods. By default, all methods created in an interface are automatically public and abstract. Constants can be created in an interface but they are treated as final and static.

```
          public interface Trainable {
               public void speak( );
          }
```

Using an interface is similar to extending a class, except that it instead uses the keyword `implements`. Only one class may be extended, but any number of interfaces can be implemented (separated by commas), this allows Java to (sort of) do multiple-inheritance. Note: It is also possible to create polymorphic objects using interfaces by creating a reference variable that is of the interface's type.

```
     public class Dog extends Animal implements Trainable {
          ...
          @Override
          public void speak( ) {
               ...
          }
     }
```

```
/**
 *  Animal is an abstract representation of an animal
 */
public abstract class Animal {
      /** Name of the animal. */
      private String name;
      /** Initializes the animal using input parameters
       *  @param n    the name of the animal.
       */
      public Animal( String n ) {
            name = n;
            doBehavior(1);
      }
      /** Accessor for the animal's name.
       *  @return the name of the animal
       */
      public String getName() {
            return name;
      }
      /**
       * Causes the animal to perform a behavior
       * @param type choose a type of behavior
       * (if there is a choice)
       */
      public abstract void doBehavior( int type );
}
```

```java
/** Interface for trainable behaviors of an object */
public interface Trainable {
     /** Method to make the object speak */
     public void speak();
     /** Method to make the object sit */
     public void sit();
     /** Method to make the object fetch */
     public void fetch();
}

/** Cat is a representation of a cat, subclass of Animal */
public class Cat extends Animal {
     /** Initializes the cat using input parameters
      *  @param n    The name of the cat.  */
     public Cat( String n ) {
          super( n );
     }
     /** Makes the cat take a nap */
     public void sleeps() {
          System.out.println(getName()+" sleeps in a sunbeam.");
     }
     /**Cats only perform a single behavior
      * @param type not used */
     @Override
     public void doBehavior( int type ) {
          sleeps();
     }
}

/** Dog is a representation of a dog, subclass of Animal */
public class Dog extends Animal implements Trainable {
     /** Initializes the dog’s name
      *  @param n    The name of the dog. */
     public Dog( String n ) {
          super( n );
     }
     /** Makes the dog speak */
     @Override
     public void speak() {
          System.out.println( "Arf Arf" );
     }
     /** Makes the dog sit */
     @Override
     public void sit() {
          System.out.println(getName()+ " sits on the ground.");
     }
     /** Makes the dog fetch */
     @Override
     public void fetch() {
          System.out.println("You throw the stick.\"Get it!\"");
          System.out.println(getName() + " fetches the stick.");
     }
```

```
	/**
	 * Dog performs a behavior given a behavior type
	 * @param type 1=speak, 2=sit, 3=fetch
	 */
	@Override
	public void doBehavior( int type ) {
		if( type == 1 ) {
			speak();
		} else if( type == 2 ) {
			sit();
		} else if( type == 3 ) {
			fetch();
		}
	}
}

/** Test implemenation of Animal and Trainable */
import java.util.Scanner;
public class AnimalTraining {
	public static void main(String[] args) {
		Animal [] animals = new Animal [2];
		animals [0] = new Cat( "Fluffy" );
		animals [1] = new Dog( "Spot" );
		int animal = animalMenu();
		int trick = trickMenu();
		animals[animal-1].doBehavior( trick );
	}
	/** Method to display Animal menu */
	public static int animalMenu() {
		Scanner in = new Scanner(System.in);
		System.out.println( "Animal Menu" );
		System.out.println( "1. Cat" );
		System.out.println( "2. Dog" );
		return in.nextInt();
	}
	/** Method to display trick menu */
	public static int trickMenu() {
		Scanner in = new Scanner(System.in);
		System.out.println( "Trick Menu" );
		System.out.println( "1. Speak" );
		System.out.println( "2. Sit" );
		System.out.println( "3. Fetch" );
		return in.nextInt();
	}
}
```