Q1 [A question from a previous CS2312 exam]

(A) Complete the following program by implementing required methods/ classes / interfaces. The expected outputs of the program are given as underlined contents in the inline comments in **`main()`**.

(B) Using any code in part (A), explain the terms Up-casting and Dynamic Binding.

```
public class Main {
    public static void main(String[] args) {
        Decider d1 = new WeatherDecider();
        Decider d2 = new BudgetDecider();
        goHiking(d1); //Output: Go hiking only if no rain.
        goHiking(d2); //Output: Go hiking anytime!
        goShopping(d1); //Output: Go shopping any time!
        goShopping(d2); //Output: Go shopping if we have money.
    }

    //Add the methods: goHiking and goShopping









}
```

```
// Add the code for Decider and WeatherDecider
// (The code for BudgetDecider is optional)
```

Q2. Cook and Food [A question from a previous CS2312 exam]

(a) Complete the given program by implementing all required classes / interfaces. The expected outputs of the program are given as underlined contents in the inline comments in **`main()`**.

Write all remaining classes/interfaces:

Given:

```
public class Main
{
   public static void main(String[] args)
   {
      Cook c1 = new SteamCook();
      Cook c2 = new CurryCook();
      c1.cook(new Fish());          //Output: steamed fish
      c2.cook(new Chicken());       //Output: chicken in curry
      c1.cook(new Chicken());       //Output: steamed chicken
      c2.cook(new Fish());          //Output: fish in curry
   }
}
```

```
interface Cook {
    void cook(Food f);
}
```

(b) Mary thinks that the **Cook** interface could be turned to a more useful class as on the right:
Briefly discuss how Mary's thought satisfies or violates the OCP (*Open-Closed Principle*)

```
public class Cook {
  private int cooking_type; /* cooking type: 1 - steam, 2 - curry */
  public void cook(..) {
       .. /* depends on the cooking type, output the result */
  }
}
```

Q3 Complete the program below by writing the classes "Do", "Re", "Mi", "Fa".

[Questions borrowed from Dr Sam NG CS2332 / 2010-11 Sem B]

Notes: You cannot modify the given code. You may write additional helper classes.

```
public class Sentence {

    public void print() {System.out.println("!!!");}
}

-------------------------------------------------------------------

public class Main {

    public static void main(String [] args)
    {
        Sentence[] s = new Sentence[3];
        s[0] = new Do(new Re(new Mi(new Sentence())));
        s[1] = new Do(new Mi(new Do(new Mi(new Sentence()))));
        s[2] = new Re(new Mi(new Fa(new Fa(new Mi(new Re(new Fa(new Sentence()))))));

        for (int i=0; i<3; i++)
            s[i].print();

    }
}
```

Here is the expected output of the program:

```
do re mi !!!
do mi do mi !!!
re mi fa fa mi re fa !!!
```

Q4 Complete the program below by writing all necessary classes.

[Questions borrowed from Dr Sam NG CS2332 / 2010-11 Sem B]

Notes:
1. You cannot modify the given code.
2. You may write additional helper classes wherever appropriate.
3. You cannot check the types of objects (eg using *instanceof*); otherwise you will get 0 marks.

```
public class Animal {
    private AgeGroup group;
    private Type type;

    public Animal(Type type, AgeGroup group) {
        this.type = type;
        this.group = group;
    }

    public void call() {
        group.call(type);
    }
}

-------------------------------------------------------------------------

public class Main {

    public static void main(String [] args)
    {
        Animal[] pets = new Animal[6];
        pets[0] = new Animal(new Dog(), new Baby());    // A baby dog is called a puppy.
        pets[1] = new Animal(new Dog(), new Adult());
        pets[2] = new Animal(new Cat(), new Baby());    // A baby cat is called a kitten
        pets[3] = new Animal(new Cat(), new Adult());
        pets[4] = new Animal(new Rabbit(), new Baby());  // A baby rabbit is called a bunny.
        pets[5] = new Animal(new Rabbit(), new Adult());

        for (int i=0; i<6; i++)
            pets[i].call();
    }
}
```

Here is the expected output of the program:

```
Puppy
Dog
Kitten
Cat
Bunny
Rabbit
```