Consider the following classes:

```java
abstract class Candy {
    String name;
    boolean isHealthy;
    boolean hasNuts;
    Candy(String name) {this.name = name;}
    public String toString() {
        String result = name + " by " + getBrand();
        if (hasNuts) result += " may cause allergic reaction ";
        return result;
    }
    abstract String getBrand();
}

class MMs extends Candy {
    MMs(String color) {
        super(color + " M&Ms");
        isHealthy = true;
    }
    public String getBrand() { return "Mars"; }
}

class PeanutMMs extends MMs {
    PeanutMMs(String color) {
        super(color + " peanut");
        hasNuts = true;
    }
    public String getFlavor() { return "all the same"; }
}

class ChocolateFrogs extends Candy {
    ChocolateFrogs() {
        super("Frog");
    }
    public String getBrand() { return "Magic"; }
    public String toString() {
        return super.toString() + " that runs away";
    }
}

class PowerBar extends Candy {
    PowerBar(String name, boolean healthy) {
        super(name);
        this.isHealthy = healthy;
        hasNuts = true;
    }
    public String getBrand() {
        if (isHealthy) return "Diet"; else return "Tasty";
    }
    public String toString() {
        return super.toString() + " and is overpriced";
    }
}
```

1. Draw a class hierarchy for the given classes.

```
Candy
  +--- MMs
     |   +--- PeanutMMs
     |       +--- PowerBar
  +--- ChocolateFrogs
     +--- PowerBar
```

2. In the following code, put an "X" to the left of each line that would result in a compilation error, and put two stars ("**") to the left of each line that would compile but would result in a runtime error. In considering each line, assume that all the correct lines above it have executed.

```java
// M!.- ~ -f" /. Candy a = new Candy("Tootsie Roll");
// Candy b = new MMs("brown"); b.getFlavor();
// (PeanutMMs) b).getFlavor();
// ChocolateFrogs c = new ChocolateFrogs();
// (c.getFlavor();
// (PeanutMMs) c).getFlavor();
// MMs d = new PeanutMMs("red");
// d.getFlavor();
// (PeanutMMs) d).getFlavor();
// PeanutMMs e = new PeanutMMs("blue");
// e.getFlavor();
// (Candy) e).getFlavor();
// PowerBar f = (PowerBar) new PowerBar("TrailMix",true);
// (Candy) new PowerBar("TrailMix",true);
// (MMs) new PowerBar("TrailMix",true);
```

3. What output would be printed by the following statements?

```java
MMs red = new MMs("red");
System.out.println(red);
red M&Ms by Mars

PeanutMMs green = new PeanutMMs("green");
System.out.println(green);
green peanut M&Ms by Mars may cause allergic reaction
Candy brown = new ChocolateFrogs();
System.out.println(brown);
Frog by Magic that runs away

PowerBar power = new PowerBar("Energy",false);
System.out.println(power);
Energy by Tasty may cause allergic reaction it is overpriced
```