Listing 1: Interfaces/src/resources/Potion.java

```java
package resources;

public class Potion extends Item {

    private Item[] items;

    public Potion(final String name, final Item... items) {
        super(name);
        this.items = items;
    }

    @Override
    public float getPrice() {
        float price = 0;
        for (Item item : items) {
            price += item.getPrice();
        }
        return price * 1.2f;
    }

    @Override
    public float getPower() {
        float healingPower = 0;
        for (Item item : items) {
            healingPower += item.getPower();
        }
        return healingPower * 2;
    }

    @Override
    public float getCooldown() {
        float decayTime = 0;
        for (Item item : items) {
            if (item.getCooldown() > decayTime) {
                decayTime = item.getCooldown();
            }
        }
        return decayTime * 2;
    }

    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        builder.append("Potion: [");
        for (Item item : items) {
            builder.append(item.toString() + ", ");
        }
        builder.replace(builder.length() - 2, builder.length() - 1, "]");

        return builder.toString();
    }

}

// Klasse Potion
// ? Der Preis eines Zaubertranks ergibt sich aus der Summe der Preise der
// beinhalteten Items plus einem Aufschlag von 20%.

// ? Die Heilkraft entspricht dem Doppelten der Summe der Heilkräfte der
// beinhalteten Items.

// ? Die Abklingzeit entspricht dem Doppelten der maximalen Abklingzeit der
// beinhalteten Items.
```

```
65 // ? Diese Werte sollen nicht in Feldern gespeichert werden, sondern
   // entsprechend dem Composite-Pattern
   // in den einzelnen Gettern berechnet werden.

   // ? Überschreiben Sie des Weiteren die toString()-Methode, in der Sie "Potion:
70 // ", gefolgt von
   // super.toString(), gefolgt von einer öffnenden eckigen Klammer, gefolgt von
   // (durch Beistriche getrennt) den textuellen Darstellungen der beinhaltenden
   // Items (abgefragt über toString()) und abschließend gefolgt von einer
   // schließenden eckigen Klammer zurückgeben. Verwenden Sie zum Aufbauen dieses
75 // Strings einen StringBuilder.
```

Listing 2: Interfaces/src/resources/Item.java

```
1  package resources;

   import pricing.DiscountRate;
   import pricing.Priced;
5
   public abstract class Item implements Priced {

       protected final String name;

10     public Item(final String name) {
           this.name = name;
       }

       @Override
15     public float getDiscountPercent(final DiscountRate rate) {
           switch (rate) {
               case LOW:
                   return 0.01f;
               case MEDIUM:
20                 return 0.05f;
               case HIGH:
                   return 0.15f;
               default:
                   return 0.00f;
25         }
       }

       public abstract float getPower();

30     public abstract float getCooldown();

       @Override
       public String toString() {
           return this.name;
35     }
   }

   /**
    *
40  * Überschreiben sie getPrice() nicht, dies wird von den Kinderklassen
    * implementiert. ?
    *
    *
    * Jedes Item hat neben dem Preis o einen Namen, o eine Heilkraft Power (dh,
45  * beim Verzehr des Items wird die angegebene Menge an Lebensenergie des
    * Spielers wiederhergestellt),
    *
    * o sowie eine Abklingzeit Cooldown (dh, die Zeit, die verstreichen muss, bis
    * dasselbe Item erneut verzehrt werden kann ? angegeben in Sekunden).
50  *
```

```
 * ? Definieren Sie für die Eigenschaften passende Getter-Methoden,
 * gegebenenfalls als abstrakt, um von den Kinderklassen implementiert zu
 * werden.
 */
```

Listing 3: Interfaces/src/resources/Resource.java

```
1  package resources;
2
3  public abstract class Resource extends Item {
4
5      public Resource(String name) {
6          super(name);
7      }
8
9      @Override
10     public float getPrice() {
11         return this.name.length();
12     }
13
14 }
15
16 // Klasse Resource
17 // ? Ist eine abstrake Klasse und stellt die atomaren Teile des Kompositums dar,
18 // also jene Items, die sich aus keinen weiteren Items zusammensetzen.
19 // ? Entwickeln Sie einen Konstruktor, der den Namen der Ressource
20 // entgegennimmt.
21 // ? Der Preis einer Resource ergibt sich aus der Länge des Namens, dh, die
22 // Ressource kostet ein Geld pro Buchstabe im Namen. Beispiele: Ein ?Ei? kostet
23 // 2 Geld, ?Schnittlauch? kostet 12 Geld. Überschreiben Sie
24 // den Preis-Getter entsprechend.
```

Listing 4: Interfaces/src/resources/AnimalProduct.java

```
1  package resources;
2
3  public class AnimalProduct extends Resource {
4
5      public AnimalProduct(String name) {
6          super(name);
7      }
8
9      @Override
10     public float getPower() {
11         return this.getPrice() * 2;
12     }
13
14     @Override
15     public float getCooldown() {
16         return 10;
17     }
18
19     @Override
20     public String toString() {
21         return "AnimalProduct: " + super.toString();
22     }
23
24 }
```

Listing 5: Interfaces/src/resources/Herb.java

```
1  package resources;
2
3  public class Herb extends Resource {
```

```
    private final float healingPower;
    private final float decayTime;

    public Herb(String name, float healingPower, float decayTime) {
        super(name);
        this.healingPower = healingPower;
        this.decayTime = decayTime;
    }

    @Override
    public float getPower() {
        return this.healingPower;
    }

    @Override
    public float getCooldown() {
        return this.decayTime;
    }

    @Override
    public String toString() {
        return "Herb: " + super.toString();
    }

}
```

Listing 6: Interfaces/src/pricing/Priced.java

```
package pricing;

public interface Priced {

    /**
     * Liefert den Preis (in Einheit ?Geld?) des jeweiligen Produktes.
     *
     * @return
     */
    float getPrice();

    /**
     * Liefert den Rabattprozentsatz des jeweiligen Produktes, abhängig von der
     * übergebenen Rabattrate.
     *
     * @param rate
     * @return
     */
    float getDiscountPercent(DiscountRate rate);

    /**
     * Liefert den Standard-Rabattprozentsatz. Diese Methode soll standardmäßig den
     * Rabattprozentsatz liefern, der mit der Rabattrate DiscountRate.LOW erzielt
     * wird.
     *
     * @return
     */
    default float getDiscountPercent() {
        return getDiscountPercent(DiscountRate.LOW);
    }

    /**
     * Liefert den Rabatt (in Einheit ?Geld?), also jene Menge an Geld, die gespart
     * wird, wenn der Rabattprozentsatz der angegebenen Rabattrate angewendet wird.
     *
     * @param rate
     * @return
```

```
     */
    default float getDiscount(DiscountRate rate) {
        return getPrice() * getDiscountPercent(rate);
    }

    /**
     * Liefert den Standardrabatt (in Einheit ?Geld?), also jene Menge an Geld, die
     * gespart wird, wenn der Standard-Rabattprozentsatz angewendet wird.
     *
     * @return
     */
    default float getDiscount() {
        return getPrice() * getDiscountPercent();
    }

    /**
     * Liefert den verbilligten Preis, also jene Menge an Geld, die bezahlt werden
     * muss, wenn vom Originalpreis der Rabatt entsprechend der angegebenen
     * Rabattrate abgezogen wird.
     *
     * @param rate
     * @return
     */
    default float getReducedPrice(DiscountRate rate) {
        return getPrice() - getDiscount(rate);
    }

    /**
     * Liefert den verbilligten Preis, also jene Menge an Geld, die bezahlt werden
     * muss, wenn vom Originalpreis der Standardrabatt abgezogen wird.
     *
     * @return
     */
    default float getReducedPrice() {
        return getPrice() - getDiscount();
    }

}
```

Listing 7: Interfaces/src/pricing/DiscountRate.java

```
package pricing;

public enum DiscountRate {
    LOW, MEDIUM, HIGH
}
```

Listing 8: Interfaces/src/app/App.java

```
package app;

import pricing.DiscountRate;
import resources.AnimalProduct;
import resources.Herb;
import resources.Item;
import resources.Potion;

public class App {
    public static void main(String[] args) throws Exception {
        Item chives = new Herb("Schnittlauch", 10, 5);
        Item basil = new Herb("Basilikum", 20, 40);

        Item spiderLeg = new AnimalProduct("Spinnenbein");
        Item chicken = new AnimalProduct("Huhn");
```

```
        Item darkpotion = new Potion("Dunkeltrank", chives, spiderLeg);
        Item nightpotion = new Potion("Nachttrank", darkpotion, spiderLeg, basil,
            chicken);

        test(chives);
        test(basil);
        test(spiderLeg);
        test(chicken);
        test(darkpotion);
        test(nightpotion);
    }

    public static void test(Item itemToTest) {
        System.out.println("START OF TEST FOR ITEM");
        System.out.println("-----------------------------------------------");

        System.out.println(itemToTest.toString());
        System.out.println(itemToTest.getPrice());
        System.out.println(itemToTest.getReducedPrice(DiscountRate.HIGH));
        System.out.println(itemToTest.getReducedPrice());
        System.out.println(itemToTest.getPower());
        System.out.println(itemToTest.getCooldown());

        System.out.println("END OF TEST FOR ITEM");
        System.out.println("-----------------------------------------------");
    }
}
```