Food Orders

- Goal: write a program to calculate the price of an order at a Hamburger stand.
- An order consists of one or more items where each item can have zero or more toppings.
- Items:
  - Hamburger - $2.50
- Toppings
  - Cheese - $0.20
  - Bacon - $0.40
  - Pickles - $0.10
  - Mushrooms - $0.30
  - Tomatoes - $0.15

• Although the only item that the stand sells is hamburgers, we want to write the code so that we can easily add other items.

Orders

- orders are read from standard in (keyboard) and have the following format:

```
item
topping
topping
[empty line]
item
topping
[empty line]
[empty line]
```

•There can be any number of toppings
• The topping list is terminated by a newline
• The order is terminated by a newline

Design Questions

- What are the objects in this program?
  - hamburger
  - topping
  - order
- What is the relationship between objects?
  - An order has hamburgers (notice the plural)
  - A hamburger has toppings

Objects

- Topping
  - Data members
    - name
    - price
  - Operations:
    - get price
- Order
  - Data members
    - list of hamburgers
  - Operations
    - get price
    - print order
- Hamburger
  - Data members:
    - name
    - base price
    - toppings list
  - Operations:
    - get price

New concepts

- Input from the keyboard
  - BufferedReader, InputStreamReader, System.in
  - exceptions
- Constructors
- Vectors
  - contain objects
  - dynamic sizing
  - Methods: add(Object o), get(int index), size()
- toString
  - every object contains a toString method that we can override.
- Equals

Vectors

- A vector is a growable list of Objects
- Methods:
  - `boolean add(Object o)` // returns true
  - `int size();`
  - `Object get(int index)` // return the object at index
  - `Object remove(int index)` // remove the object at index, and return it