An interesting and useful concept is having an efficient and convenient way to map a unique data key to a specific value.

We could do this by having our own integer mapping for the data keys and then use something such as an array, but that can be limiting and unintuitive.

In Java, the Map<Key,Value> interface describes a more general purpose way of doing this, and allows for multiple implementations to be made, each with its own strengths.
EnumMap

One implementation of the `Map` interface is `EnumMap` which is meant to easily connect the items in an enumeration to specific values.

```java
public static enum Outcomes { FRESH, ROTTEN, NEUTRAL }

public static void main(String[] args) {
    Map<Outcomes,Integer> ratingToInteger;
    ratingToInteger = new EnumMap<Outcomes,Integer>(Outcomes.class);
    ratingToInteger.put(Outcomes.ROTTEN, -1);
    ratingToInteger.put(Outcomes.NEUTRAL, 0);
    ratingToInteger.put(Outcomes.FRESH, +1);
    for (Outcomes val : Outcomes.values()) {
        System.out.println(val + " " + ratingToInteger.get(val));
    }
}
```

TreeMap and HashMap

Two other implementations of the `Map` interface are `TreeMap` and `HashMap` which easily connect the unique immutable objects to specific values.

In terms of functionality, these two are interchangeable. Where they differ is “under the hood” in the way they organize the information, and their efficiency in accessing that information.
TreeMap and HashMap

Map<String,Integer> planetToSize = new TreeMap<String,Integer>();
planetToSize.put("MERCURY",1516);
planetToSize.put("VENUS",3761);
planetToSize.put("EARTH",3959);
planetToSize.put("MARS",2460);
planetToSize.put("JUPITER",43441);
planetToSize.put("SATURN",36184);
planetToSize.put("URANUS",15759);
planetToSize.put("NEPTUNE",15299);
planetToSize.put("PLUTO",737);

Scanner sc = new Scanner(System.in);
System.out.print("Enter the name of a planet: ");
String planet = sc.next().toUpperCase();
System.out.println(
        "The radius of " + planet +
        " is approximately " + planetToSize.get(planet) +
        " miles."
    );

System.out.println("Contents: " + planetToSize.keySet());

CMSC132

One of the things that you will explore more in CMSC132 is the topic of data structures and scenarios where they are best used. Maps and hash tables and trees will be among these.
Copyright © 2017 : Evan Golub