ITEC 120

Lecture 26

Data storage with objects

Review

- Questions
- Paradigm shift
- What syntax do we use for creating objects?

Objectives

- Focus on storing data in objects
- Initialization
- Protecting integrity of data

Questions

- What types of data do computer programs store?
- Why is data stored in computer programs?
- How is data manipulated in computer programs?
- How is data reported in computer programs?

Refresher

- Grouping data values together
  - Specify more than one variable

```
int number;
String type;
String review;
int calories;
```



```
class Apple
{
    public int number;
    public String type;
    public String review;
    public int calories;
}
```

Procedural

Object Oriented

OO Data

Usage

Procedural

```
int number;
String type;
String review;
int calories;

number = 4;
type = "Red";
review = "Sweet and juicy";
calories=80;
```

Object oriented

```
class Apple
{
    public int number;
    public String type;
    public String review;
    public int calories;
}

Apple basket = new Apple();
basket.number=4;
basket.type = "Red";
basket.review = "Sweet and juicy";
basket.calories = 80;
```

OO Data

Advantage

- Scaling up

2 Apples

```
int number;
String type;
String review;
int calories;
int number2;
String type2;
String review2;
int calories2;
number=10;
number2=20;
```

```
class Apple
{
    public int number;
    public String type;
    public String review;
    public int calories;
}
Apple a = new Apple();
Apple b = new Apple();
a.number=10;
b.number=20;
```

OO Data

Mental Model

- Procedural
  - Pieces of paper on a desk
- Object oriented
  - File folder full of paper on a desk

versus

OO Data

Functions

- Passing variables

```
int number;
String type;
String review;
int calories;
funcs.munch(number,type,review,calories);
```

Lots of type required for multiple variables

```
class Apple
{
    int number;
    String type;
    String review;
    int calories;
}
Apple a = new Apple();
funcs.munch(a);
```

Only one parameter required

Functions

- Receiving variables
  - Procedurally

```
public void munch(int number, String type, String review,
                  int calories)
```

  - Object Oriented

```
public void munch(Apple toEat)
```


Modifications

- Procedurally
  - Values in calling function do not change
- Object oriented
  - Values in calling function do change
- Cannot change where object is pointing though
- Code example

Recall

- Classes are like arrays

```
int[] x;
```
→ References nothing

Versus

```
int[] x = new int[10];
```
References an array of size 10

```
Apple a;
```
→ References nothing

Versus

```
Apple a = new Apple();
```
→ References memory that stores values that describe an apple

File location

- Two choices
- Inside another .java
  - Removes need for separate files
  - Prevents usage of class in other files
- In separate file
  - Apple.java would hold
  - Usable by other files

```
class Apple
{
    int number;
    String type;
    String review;
    int calories;
}
public class file
{
    public static void main…
    {

    }
}
```

```
public class Apple
{
   public int number;
   public String type;
   public String review;
   public int calories;
}
```


Initialization

- Default values procedurally

```
int number;
String type;
String review;
int calories;
```

By default you cannot guarantee what these variables hold.

```
number = 4;
type = "Red";
review = "Sweet and juicy";
calories=80;
```

Solution 1

```
int number=4;
String type="Red";
String review="Sweet and juicy";
int calories=80;
```

Solution 2

Initialization

- Object oriented

```
class Apple
{
   int number;
   String type;
   String review;
   int calories;
}

Apple basket = new Apple();
basket.number=4;
basket.type = "Red";
basket.review = "Sweet and juicy";
basket.calories = 80;
```

Or you can leave out the next 4 lines

Constructors

- Function called when an object is created
- Same name as the class
- No return type
- Takes as many parameters as you want

Constructor

```
Apple basket = new Apple();
```

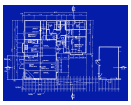
Example

```
class Apple
{
    public int number;
    public String type;
    public String review;
    public int calories;
    public Apple()
    {
        number=4;
        type = "Red";
        review = "Sweet and juicy";
        calories = 80;
    }
}
Apple a = new Apple();
System.out.println(a.type);
```

What does this print?

Reminder

- A class is a description not an instance
- Variables are instances not descriptions



versus

Coding

- Soda machine that can hold 6 types of soda
- Starts out fully stocked
- jEdit example

Review

- Object oriented data storage
- Syntax
- Constructors!
- How it works with functions
- File location
- Comparison to regular variables