RADFORD UNIVERSITY

ITEC 120

Lecture 27
Protecting data

Review

- Questions?
- Objects and data
  - Difference from regular variables
- Constructors

Objectives

- How to protect your data
- Mixing variables and functions

Questions

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

- What happens when
  - A negative number of apples is set?
  - A typo is made in the type of apple?
  - Bad reviews are given?
  - Calories are misrepresented?

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

Motivation

- Class variables are property of the class

```
class Person
{
    public int diamonds;
    private String pizza;
}
```

Data protection

Solution

- Guarantee values are correct

Responsibility

You the programmer

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

Apple object itself

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

Data protection

How

- Program the Apple class to take care of its data

```
class Apple
{
   public int number;
   public String type;
   public String review;
   public int calories;
}
Apple a = new Apple();
a.number=4; // Will compile
```

versus

```
class Apple
{
   private int number;
   private String type;
   private String review;
   private int calories;
}
Apple a = new Apple();
a.number=4; //Will not compile
```

Data protection

Problem

- Private means no one on the outside can access the variables
- How do you access the variables?
  - Functions
- Functions in a class can use variables in the class

Data protection

Example

Paradigm shift!

Constructor
Sets default values

Setter
Sets a particular value

Getter
Makes a copy of the value

```
class Apple
{
    private int number;
    private String type;
    private String review;
    private int calories;
    public Apple()
    {
        number=calories=0;
        type=review="none";
    }
    public void setNumber(int value)
    {  if (value >0)
           number=value;
    }
    public void getNumber()
    {  return number;}
}
Apple a = new Apple();
a.setNumber(4); //Will not compile
```

Data protection

Terms

- Constructor
  - Executed once
  - Works on all variables
- Getter / Accessor
  - Makes a copy of requested resource
- Setter / Mutator
  - Sets a private value
  - Can keep a value in a certain range

Data protection

Pro/Con

- Pro
  - Ensures data values are always initialized (constructor)
  - Variable access is always monitored
  - Value can be used w/out fear of manipulation
- Con
  - More coding
  - Not as easy to use as .valueName

Data protection

Rule of thumb

- All class variables should be private
- How do you get access to them?

```
public class Example
{
    private int ring;
    Example() { ring =1;}
}
```

Not yours

Data protection

Accessor / Getter

- Makes a copy of the variable
- Gives copy to requested party

```
public class Example
{
   private int ring;
   Example()
       { ring=1;}
   int getRing()
       { return ring; }
}
```

```
public static void main(String[] args)
{
    Example a = new Example();
    System.out.println(a.getRing());
}
```

a.ring ⟷

Data protection

Mutators / Setter

- Grant access to private variables

```
class Data
{
   private int amount;
   public Data()
       { amount=20;}
   int setAmount(int newAmount)
   {
       if (newAmount > 10)
       {
          amount=20;
       }
   }
}
```

```
public static void main(
        String[] args)
{
    Data a = new Data();
    Data b = new Data();
    a.setAmount(30);
    b.setAmount(10);
}
```

Data protection

Data classes

- One acceptable use of public variables
- Container for lots of variables

```
class Op
{
    public String command;
    public String first;
    public String second;
    public Op()
    {
        command="none";
        first="none";
        second="none";
    }
}
```

```
public static void main
        (String[] args)
{
    Op a = new Operation();
    a.command="multiply";
    a.first = "red";
    b.second="white";
    performOp(a);
}
```

Data protection

Decisions

- Prim and proper
  - All variables private
- Fast and loose
  - Everything public

Data protection

Encapsulation

- Walled gardens of safety
- Keep other classes / drivers out
- Make assumptions about variables

Data protection

Process

- Determine what information should be grouped together
- Decide whether or not values must always stay in a certain range
- Decide who should have access to what variables
- Are copies of value desirable, or the original?

Data protection

Review

- Encapsulation
- Syntax
- Examples

Data protection