Classes, Objects, and Methods

• Class: a definition of a kind of object
• Object: an instance of a class
  – Contains instance variables (data) and methods
• Methods
  – Methods that return a value
  – Methods that return nothing
Variable Scope

1) After declaration
2) Within the code block of declaration

Correct:
```java
int i;
for ( i = 100; i<200; i++ ) {
    if ( i % 67 == 0 ) break;
}
System.out.println(i);
```

Wrong:
```java
for ( int i = 100; i<200; i++ ) {
    if ( i % 67 == 0 ) break;
}
System.out.println(i);
```
Local/Instance variables

• Instance variables
  – Declared in a class
  – Confined to the class

• Local variables
  – Declared in a method
  – Confined to the method

```java
public class Student {
    public String name;
    public int classYear;
    public String major;

    public void printInfo() {
        String info = name + ":" + major + ":" + classYear;
        System.out.println(info);
    }

    public void increaseYear(int inc) {
        classYear += inc;
    }
}
```
Simple example

```java
public class Student {
    public String name;
    public int classYear;
    public String major;

    public void printInfo() {
        String info = name + "\: " + major + "\: " + classYear;
        System.out.println(info);
    }

    public void increaseYear(int inc) {
        classYear += inc;
    }
}
```

- `classYear` and `name` are instance variables
- can be used in any method in this class

- `info` is a local variable declared inside method `printInfo()`
- can only be used inside method `printInfo()`
public class Student {
    public String name;
    public int classYear;
    public String major;

    public void printInfo() {
        String info = name + ": " + major + ": " + classYear;
        System.out.println(info);
    }

    public void increaseYear(int inc) {
        classYear += inc;
        info = "info changed a bit";
    }
}

• Java will not recognize info
public class Student
{
    public String name;
    public int classYear;
    public String major;

    public void printInfo()
    {
        String info = name + " : " + major + " : " + classYear;
        System.out.println(info);
    }

    public void increaseYear(int inc)
    {
        classYear += inc;
        String info = "classYear updated";
        System.out.println(info);
    }
}

• The two info will not affect each other.
Methods with parameters

• Compute the square of this number
  – 5
  – 10
  – 7

• I could give you any number, and you could tell me the square of it

• We can do the same thing with methods
Methods with parameters

• Parameters are used to hold the value that you pass to the method

• Parameters can be used as (local) variables inside the method

```java
public int square(int number) {
    return number * number;
}
```
public class Student {
    public String name;
    public int classYear;
    // ...

    public void setName(String studentName) {
        name = studentName;
    }

    public void setClassYear(int year) {
        classYear = year;
    }
}
Calling a method with parameters

Student andrew = new Student();
andrew.setName(“Andrew Matthews”);
andrew.setClassYear(3);

The type of argument must match the type of parameters as defined in method header

```java
public void setName(String studentName)
public void setClassYear(int year)
```
Methods with multiple parameters

• Multiple parameters separated by commas

```java
public class SalesComputer {
    public double getTotal(double price, double tax) {
        return price + price * tax;
    }
}
```

• **Order, type, and number** of arguments must match parameters specified in method header

```java
SalesComputer sc = new SalesComputer();
double total = sc.getTotal("19.99", Color.RED);  // X
double total = sc.getTotal(19.99);
double total = sc.getTotal(19.99, 0.065);
int price = 50;
total = sc.getTotal(price, 0.065);
```
Worksheets

- Chapter 1 & 2
- String & loop
Next Class

- Lab 5
- Bring laptop & textbook