Java without Greenfoot

- No animation, no world that runs act() methods of all objects
- Instead we write a client program (class) with a "main" method to run our project (scenario)
- Interface to programs is text based prompting for input and display of output. Also file reading/writing.

Classes of Objects

*What about multiple objects with the same properties?*

**Class**

A specification of the representation of a particular kind of object, in terms of data and behavior (operations)

Object-Oriented Problem Solving

- Brainstorm - List all objects that might contribute to solution
- Filter - Review the classes to find duplicates or remove unnecessary objects
- Responsibilities - Determine the operations associated with a class of objects
- Class Relationships

Class Relationships

- Collaboration – classes that work together using message passing
- Composite – a class that contains other classes. "has a" relationship
- Inheritance – a class that is a subclass of another. "is a" relationship

Problem-Solving Techniques

- Attributes - Determine the values defined by a class that are used to represent its objects
- Methods - Support the responsibilities and collaborations
- Driver - Write a program for creating the objects and coordinating their collaborations to solve the problem
Class Design Principles

- Encapsulation - Designing a class so that its attributes are isolated from the actions of external code except through the formal interface
- Formal interface (API) - The components of a class that are externally accessible, which consist of its non-private fields and methods

Class Design Principles

- Abstraction - The separation of the logical properties (interface and specification) of an object from its implementation (key to handling complexity)
- Modifiability - The property of an encapsulated class that allows the implementation to be changed without having an effect on code that uses it

Class Design Principles

- Reuse - The ability to import and use a class in different contexts without requiring modifications to either class or using code

Some Terminology

- Object reference: identifier of the object
- Instantiating an object: creating an object of a class
- Instance of the class: the object
- Methods: the code to manipulate the object data
- Calling a method: invoking a service for an object.

Date class example

Date class
- Instance fields: month, day, year
- Instance Methods: set or get month, day, year; leap year?, increment
day/week/month/year, dayOfWeek?

Date object (an instance of a class)
- June
- 23
- 2004

Class methods/attributes

- Class method A method that belongs to a class rather than its object instances; has modifier static
  Date.setDefaultFormat(Date.MONTH_DAY_YEAR);
- Class field A field that belongs to a class rather than its object instances; has modifier static
What is a Method?
- Named piece of code to perform a specific task
- Called with optional arguments from somewhere else in your program
- When called, program control (Execution) is transferred to the method
- Method performs required tasks, and then possibly returns a value (or values or object)
- After return from the method, control returns to the statement following the method call
- See the Class API for method details

Method Arguments
- Methods can have as many arguments as necessary, each expressed as pairs `dataType variableName`
- Methods specify the order and data type of arguments
- When calling a method, arguments can be any expression that evaluates to the specified data type
- A Copy of the Value of the Parameter Is Passed to the Function
- Changing the Value of the Parameter in the Function Does Not Affect the Value of the Original Variable
- This Is Called Pass-by-Value

Method Return Values
- No return value is required
- Can be a primitive data type, class type, or `void`
- A value-returning method
  - Return value is not `void`
  - The method call is used in an expression. When the expression is evaluated, the return value of the method replaces the method call.
- Methods with a `void` return type
  - Have no value
  - Method call is complete statement (ends with ;)

Calling Methods
- You need to match the data types and number of arguments in the method definition (do not include data types when calling)
- Arguments can be values, variables or expressions (of the correct data type)
- If the method takes no arguments, remember to include the empty parentheses after the method's name. The parentheses are required even if there are no arguments.

Dot Notation
- Use when calling method to specify which object's data to use in the method

```
objectReference.methodName( arg1, arg2, ... )
```

Note: no data types in method call; values only!