Components of user-defined classes

- Data: variables
- Action: methods
- Initialization: constructor

Example: Bibliography items
(Friday's example)

- We started by determining what data we needed to represent the different items:
  - author, title, publisher, ...
- We then determined what “actions” the class needed to operate on the data:
  - updateAuthor(...), updateTitle(...), print(), ...

Example: Class Book

```java
public class Book {
    private String author;
    private String title;
    private String date;
    private String publisher;
    ...
    public void print() {
        System.out.println(author + " title ", "date ");
    }
    public void updateAuthor(String name) {
        author = name;
    }
    ...
}
```
The third item: the Constructor

- A constructor is a special method used to initialize an instance of a class
- We use the constructor for a class to initialize the key variables for the class
  - i.e., the data that's necessary for the methods in the class to operate correctly
- A class can have more than one constructor
  - more on this later
- Constructors do not return a value

Example: Class Book

```java
public class Book {
    ...
    public Book(String name, String bookTitle, String bookPublisher, String publishDate) {
        author = name;
        title = bookTitle;
        publisher = bookPublisher;
        date = publishDate;
    }
    ...
}
```

More examples

- Q: What would the constructor for the class you wrote on Friday look like?

Multiple constructors

- Sometimes you want to initialize different data in particular cases when creating an object of a class
- To do so, you can define more than one constructor for the class
  - each constructor must have a different number of parameters, OR different types for the parameters
Example: Class Book

```java
public class Book {
    ...
    public Book() {
        author = title = publisher = date = "";
    }
    public Book(String name, String bookTitle, String bookPublisher, String publishDate) {
        author = name;
        title = bookTitle;
        publisher = bookPublisher;
        date = publishDate;
    }
    ...
}
```

Information hiding

- Idea: We don't want everyone to be able to access everything in our classes
  - typically, we don't want people to be able to change the class data
  - example: currencyConverter (in the book): we don’t want someone arbitrarily changing the transaction fee
- Use visibility modifiers and accessor/mutator methods to allow the proper level of control to users

Visibility modifiers

- 3 keywords:
  - public: accessible to everyone, inside and outside the class
  - private: accessible only within the class
  - protected: accessible within the class and any subclasses of this class
- No modifier: accessible to any class within the same directory as this class
- This class: public and private

In general....

- Data members of a class are declared private
- Methods of a class are declared public
- Constructors are declared public
### Accessor and mutator methods

- Allow access to data members of a class
- Why not just declare data public?
  - allows the programmer (you) to control which data members can be accessed, and under which conditions
- Example:
  ```java
  int foo;
  public int getFoo() { return foo; }
  public void setFoo(int newFoo) { foo = newFoo; }
  ```

### Local variables

- A variable defined within a method, to be used only within that method, is a *local variable*
- Local variables only exist within the method in which they are defined
- Example: `sum` is a local variable in this method
  ```java
  public int add3Numbers(int a, int b, int c) {
    int sum = a + b + c;
    return sum;
  }
  ```

### Parameters in methods

- Q: How are parameters passed to a method, and how does the method know which parameters are which?

### More about parameter passing

- Parameters can be primitive data types, like integers, or objects
- Primitive data types are passed *by value* (i.e., the actual value is passed to the method)
- Objects are passed *by reference* (i.e., the address of the object is passed to the method)
- Methods should not change the values of parameters
  - will be lost when the method returns to its caller
Return values

- Methods have *return values*
  - indicates the type of data the method returns
- Example: public int Foo(...) returns an integer
- Example: public String Foo2(...) returns a string
- The word *return* must appear in the method
  - public int getFoo() { return foo; }
- Example: public void Foo3(...) does not return anything