3-0: Objects

Java is an object-oriented language.
So what the heck is an object anyway?

3-1: Objects

- An object is a type of abstraction
- It provides a way of grouping together related data and functionality.
- Makes it easier to organize and extend your program.
- Also gives a "black box" effect.
  - Users of your objects don't need to worry about how they work internally, just how to use them.

3-2: Classes and objects

- A class is a template or category
- An object is a particular instance of that class.
  - CartoonAnimals might be a class
  - BugsBunny, Tweety are instances of that class
- Classes let us specify behavior common to a set of objects.

3-3: An example

- Say we’re making a drawing program, and we need to represent a collection of circles.
- A circle has an x and y coordinate for its center, plus a radius.
- We could do:
  ```java
  c1xval = 3;
c1yval = 4;
c1radius = 10;
c2xval = 5;
c2yval = 2;
c2radius = 6;
...```
- (to be less messy, we could also store these in arrays)

3-4: An example

- This is not a good solution, though.
- No grouping of a circle’s components.
- Users need to know all about the internals of a circle.
3-5: An example

- A better solution:
  ```java
  public class circle
  {
      public int xval;
      public int yval;
      public int radius;
  }
  ```
- We have **encapsulated** the center and radius information inside the circle.

3-6: An example

- But we have two related variables representing the center.
- Perhaps we should group them as well.
  ```java
  public class point
  {
      public int xval;
      public int yval;
  }
  public class circle
  {
      public point center;
      public int radius;
  }
  ```

3-7: Methods

- As we know, classes also contain methods.
- Methods are pieces of code that can be invoked on an object.
- The allow us to encapsulate both **state** and **behavior**.

3-8: Data hiding

- It's also important to protect instance data from outside users.
- One way to do this is by providing **accessors** and **mutators** "setters and getters"
- Rather than the user modifying your object's data directly, they use a method to do it.
  - Reduces error
  - Hides implementation from the user.

3-9: Visibility modifiers

- public and private are used to indicate who can access a variable or method.
  - public: anyone can use it.
  - private: only available within that object.

3-10: Visibility modifiers

- Instance variables should be made private unless there's a compelling reason not to.
  - Use accessors and mutators to access and change data
- public methods are available for everyone to use.
- private methods can be used only within the object.
  - These are nice for "helper" methods that you don't want a user to call.
3-11: Example

- Design a book class.
- It should have instance variables for title, author, genre, publishers, copyright date. They should be strings.
- It should also have getters and setters for each of these.

3-12: Example

- Now create a Name class. It should have two members: firstName and lastName.
- Modify Book so that author is of type Name.
- Since you used setters and getters, users of your book class will never know!