OOP Review

Aaron Friesen

Objects

- Have both state and function
  - State - what makes the object *different* from others of its type (adjectives, nouns)
  - Function - what the object can *do* (verbs)

Inheritance/Polymorphism

- Inheritance
  - Defines a hierarchy of class relationships using the **“is-a”** ideology. A Dog **is an** Animal, a Car **is a** Vehicle, etc.
  - Allows child classes to inherit code, reducing logical redundancy and making later code changes simpler and easier
- Polymorphism
  - Technically means “having many forms”, but how useful is that?
  - Really means that an object of a child class can be assigned to the variable type of its parent class.
  - Dynamic Binding - at runtime, the most specific method possible will be invoked, based on what the **object type** of the variable is.

And finally, Enums

- Enums, or enumerated types, allow us to define a small set of values.
- Variables of that enum can **only** contain those values!
- Useful when we have a finite number of options, but they are all logically related
- Cars might have a GasType enum, which could have four values: {REGULAR, DIESEL, ELECTRIC, HYBRID}
- By convention, enum types are usually uppercase(like constants).

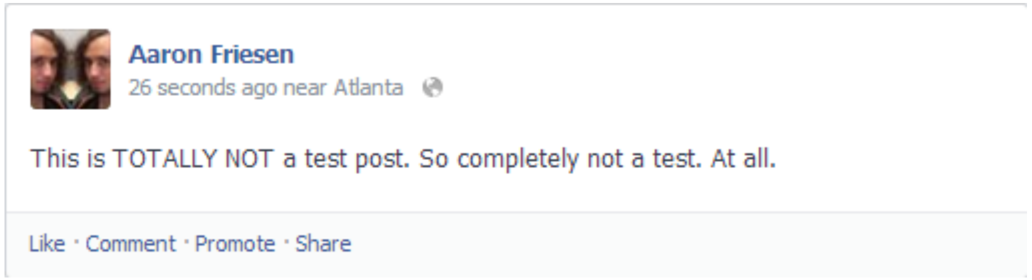
A word of warning...

- Enums can have variables associated with them, and even constructors and methods
- Enums can also extend other Enums (but not anything else)
- However, this is **nearly always** bad practice. For data that is different for different enums and never changes, this is fine. But generally try to shy away from this.

Objects are all around us!

- OOP is used in nearly every application made since the early 1990’s, with the rise of C++ and Java
- Examples

Facebook

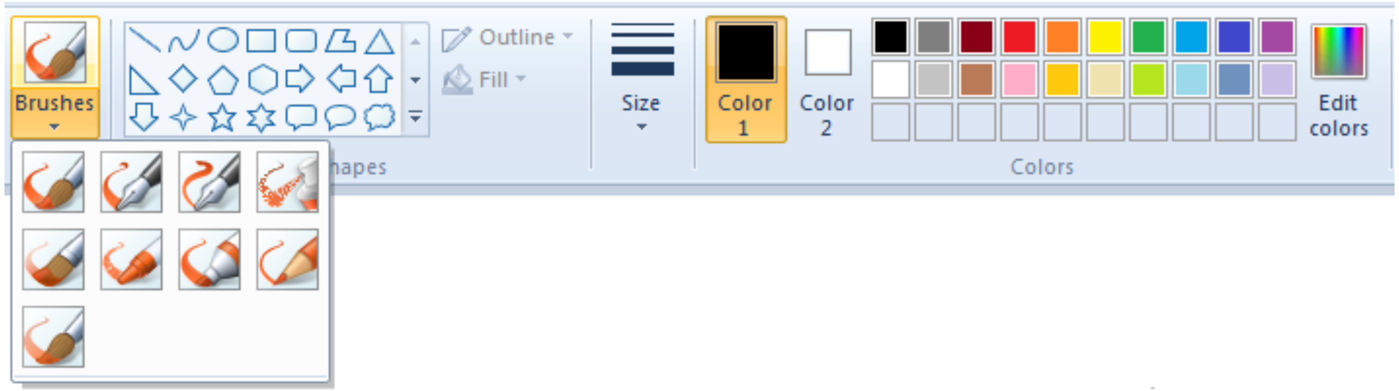


Aaron Friesen
26 seconds ago near Atlanta

This is TOTALLY NOT a test post. So completely not a test. At all.

- If you could see the code for a “FBPost” object, what variables do you think it would have?
- What methods?

Drawing Programs



- Imagine - Drawable abstract class, with Stroke and Shape subclasses
- Variables?
- Methods?

Pokemon

- TONS of detail in these.
- What variables might a Pokemon have? What kind of hierarchy? Methods?

Real-life examples

- By thinking with inheritance and OOP in mind, we can imagine superclass/subclass relationships with all kinds of real-life objects.
- SodaMachine extends VendingMachine, Guitar extends StringInstrument, etc