CS101 — Professor Anasse Bari
Prepared by Christopher Davidson
04/10/2023

TMNT OOP

Cowabunga Dudes and Dudettes!

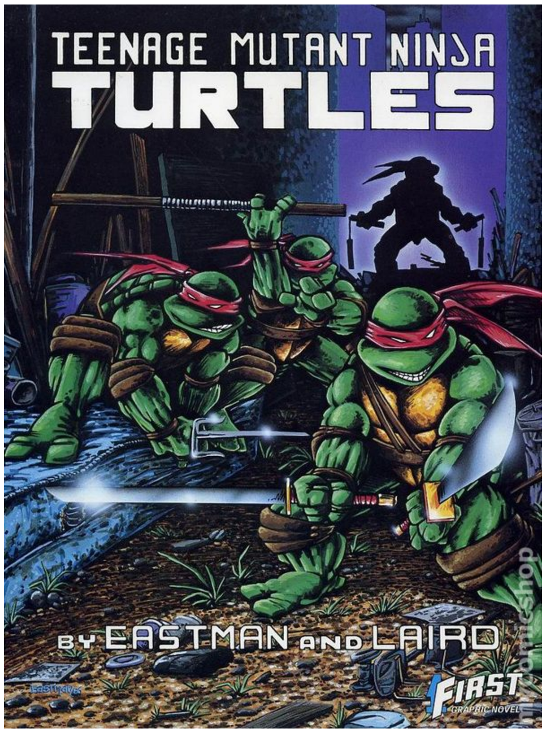
For this OOP exercise, we are going to learn all about Inheritance, the **super** keyword, and components through the use of the Teenage Mutant Ninja Turtles — or Teenage Mutant Hero Turtles if you're in Europe.

PHASE ONE: OG Turtles — NinjaTurtle.java

Originally created as a joke in 1983 by cartoonists Kevin Eastman and Peter Laird, the Teenage Mutant Ninja Turtles have gone through many incarnations throughout the years, with essentially each generation getting its own iteration of the TMNT.

First published in 1984, the original Teenage Mutant Ninja Turtles consisted of 4 members, all of whom were named after Renaissance Masters — Leonardo, Raphael, Donatello, and Michelangelo.

This initial turtle incarnation wasn't wisecracking jokes, nor dancing with Vanilla Ice. Nay, the original turtles were surprisingly brooding and serious. Oddly enough, you couldn't even tell the turtles apart, as they all wore red bandanas:



Your first task is to implement these original NinjaTurtles in a class, **NinjaTurtle.java.**

Each NinjaTurtle has a Name (String), MaskColor (String), and YearOfOrigin (int). Make these data members **protected**, please.

For Constructors, each NinjaTurtle should have the following:

1. A Default Constructor representing a “null” or “blank” turtle.
2. A Constructor that takes in a Name, MaskColor, and YearOfOrigin.
3. A Constructor that just takes in a Name.

In terms of methods, each NinjaTurtle has the following methods (and perhaps more!) :

1. Standard getters and setters
2. A **toString()** method that returns a string representation of this NinjaTurtle object.
3. The method **ninjitsu()**, which will print out one of the many wise sayings from Master Splinter. For example, feel free to choose any of the following (or figure out a way to randomly print one, time permitting):

"*If you look for happiness outside of yourself, you'll never find it. Happiness exists only within you.”*

"*Possess the right thinking. Only then can one receive the gifts of strength, knowledge, and peace.*"

"*Tonight you have learned the final and greatest truth of the Ninja: that ultimate mastering comes not from the body, but from the mind. Together, there is nothing your four minds cannot accomplish.”*

"*Running into battle without knowledge or preparation is foolish. Sometimes it is best to sit still. The answers will come.*"

Provide an accompanying class, **NYC.java**, which will contain your main() method. Make an array of NinjaTurtles, and populate it with 4 (four) NinjaTurtles, each one representing a member of the original 1984 incarnation. Print out each NinjaTurtle’s information, and have each of them perform one act of ninjitsu as well.

Specifically, you should use the 3rd Constructor (i.e., the Constructor that only takes in a name) to call the 2nd constructor with the other information you’ll need for this original team.

PHASE TWO: Heroes in a HalfShell — TMNT.java

Because there is something of an aesthetic dissonance between the name "Teenage Mutant Ninja Turtles" and their original gritty nature, the Turtles received a much-needed rebranding in 1987.

What was once black-and-white and grim all over was now replaced with jokes, hijinks, and most importantly, color-coded masks with personalized belts so that you could actually tell each turtle apart:

We are going demonstrate this re-branding in Java through the use of Inheritance.

Your task is to implement **TMNT.java.** This class will extend NinjaTurtle.java, as every TMNT is still a Ninja Turtle. You are going to make four TMNTs, representing Leo, Raph, Mikey, and Donnie (notice how even their names have become less serious).

In addition to their inherited data members, each TMNT has a **belt (char)**, representing the first letter of their names (i.e., 'L', 'R', 'M', or 'D'), and a boolean value, **lovesPizza** (set this to TRUE).

Each TMNT has the following constructors:

1. Default Constructor
2. A Constructor that takes in a Name and a MaskColor.

Use the Second Constructor to call one of the parent class' constructors using the appropriate information. Depending on the Name and MaskColor, you should also set the TMNT's belt data member as well.

In addition to having access to all of the NinjaTurtle's original methods, and standard getters and setters, the TMNT class also has the following new methods:

ninjitsu(): We need to get some more hijinks in this Java class, so let's replace the wise teachings of Master Splinter with a quote from the official theme song of the second Ninja Turtles movie, Vanilla Ice's "Go Ninja Go Ninja, Go.":

"YO. It's the green machine, going to rock the town without being seen — have you ever seen a turtle get down?"

"Gonna rock— and roll the place with the power of the ninja turtle bass".

Cowabunga(): Use this method to first call ninjitsu(), then print out "COWABUNGA!" And/or "RADICAL!" (Your choice).

Keep in mind though, I don't want you to start Cowabunga with the Vanilla Ice version of ninjitsu(), but to call upon the original ninjitsu teachings of Master Splinter.

(But Chris, how can we call upon the original ninjutsu() implementation? You tell me.)

In your **NYC.java** class, now make an array of 4 TMNT objects, with each object representing a member of the TMNT. Have each TMNT perform cowabunga, and then print them out.

Question: Which toString() method will be called when you S.O.P(TMNT)? Should you make a new toString() in TMNT?

PHASE THREE: What's Missing? ~~Weapons~~ Components!

In 1990, The Teenage Mutant Ninja Turtles were one of the first comic books to receive a film adaptation in the eponymous *Teenage Mutant Ninja Turtles*.

Despite using the reimagined TMNT aesthetic, the plot of the film actually followed some of the major arcs from the original 1984 comic. So, you got the best of both worlds: color-coordinated action with tons of ninja weapons and gritty fight scenes galore.

Parents complained however, so for 1991's *Teenage Mutant Ninja Turtles II: The Secret of the Ooze*, the ninja weapons were permanently holstered.

We're not babies though, so let's give these Ninja Turtles some weapons!

As a matter of fact, for the original 1984 incarnation, the only way to tell the turtles apart was based on what weapons they were using: Leonardo has two katanas, Raphael has two sai, Donatello has a singular bo-staff, and Michelangelo has two nunchaku (or one grappling hook if you are in Europe).

How can we easily modify our NinjaTurtle.java class to give everyone the appropriate weapon? Should we make a data member for each weapon for each turtle, setting them all to null, and then only setting the component for the appropriate turtle?

In other words, should we make a katana data member for each turtle, set it to null, and then only generate a katana when the NinjaTurtle is Leonardo?

NO! Think about how much code we would be wasting and/or repeating!

What we want to actually do is just implement each weapon using an abstract **Weapon.java** class.

Each Weapon has the data members of **count** (integer value of either 1 or 2), and **edged** (boolean, representing whether it is an edged-weapon or a blunt-force weapon)

Then, you will have 4 child classes that specifically extend this Weapon abstract class. These four classes would be:

BoStaff.java: Count of one, not edged.
Katana.java: Count of one, edged.
Nunchaku.java: Count of two, not edged.
Sai.java: Count of two, edged.

Provide each of these child classes with their own **toString()** method as well.

Finally, see if you can't go back and modify your NinjaTurtle.java class to now give every Ninja Turtle a weapon. Go through the child classes to ensure that each Named TMNT member now also receives the appropriate weapon as well.

"The greater danger for most of us lies not in setting our aim too high and falling short; but in setting our aim too low, and achieving our mark." — Michelangelo, the Sculptor.

"Wise man say: 'Forgiveness is divine, but never pay full price for late pizza.'" — Michelangelo, the Ninja Turtle.