
TECH
ELEVATOR
Module 1, Week 6

Icebreaker

If you had to describe how you’re feeling right now as an amusement park ride, what ride are you on?

Week	Topics
Week 0	Welcome, Intro to Tools
Week 1	Variables and Data Types, Logical Branching
Week 2	Loops and Arrays, Command-Line Programs, Intro to Objects
Week 3	Collections
Week 4	Mid-Module Project
Week 5	Classes and Encapsulation
Week 6	Inheritance, Polymorphism
Week 7	Unit Testing, Exceptions and Error Handling
Week 8	File I/O Reading and Writing
Week 9	End-of-module project
Week 10	Assessment

Housekeeping

- **Extra practice:** I added coding katas to your student code repos.
- **Week 7:** there are two units in the LMS, but only one (Unit Testing) has exercises. The unit on Exception Handling only has a tutorial, but no exercises.

Inheritance

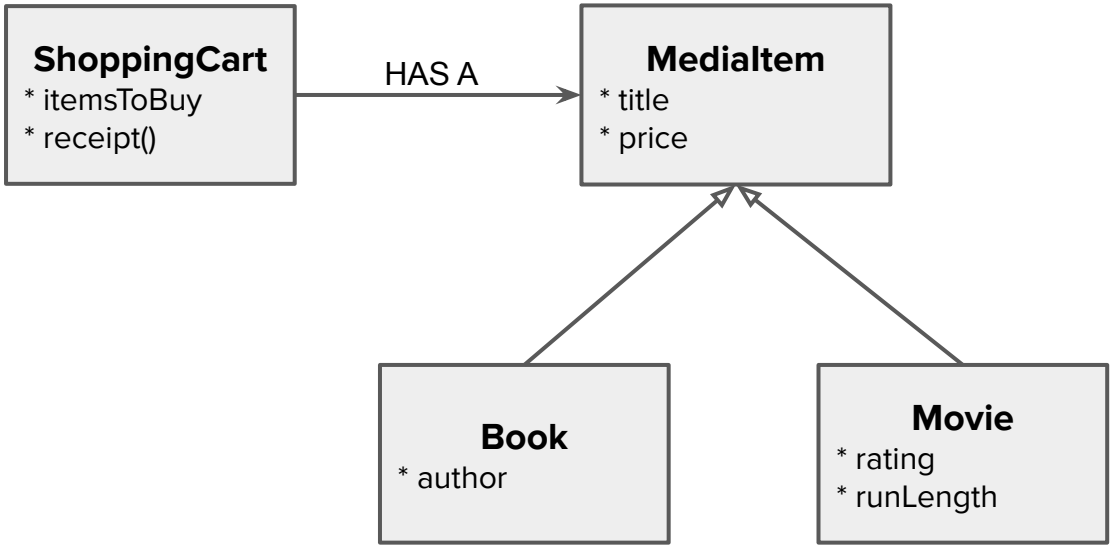
The act of having one class adopt the properties and methods of another class.

Keep things DRY - Don’t **R**epeat **Y**ourself!

What’s wrong with duplicate code?

- making a change requires extra work to update all the copies
- you might forget to update one copy, introducing (or failing to fix) a bug
- when someone else works on the code, they won't know where all the copies are
- it clutters the source code, making it harder to find other code you're looking for
- you have to duplicate your test code as well

Tutorial Example

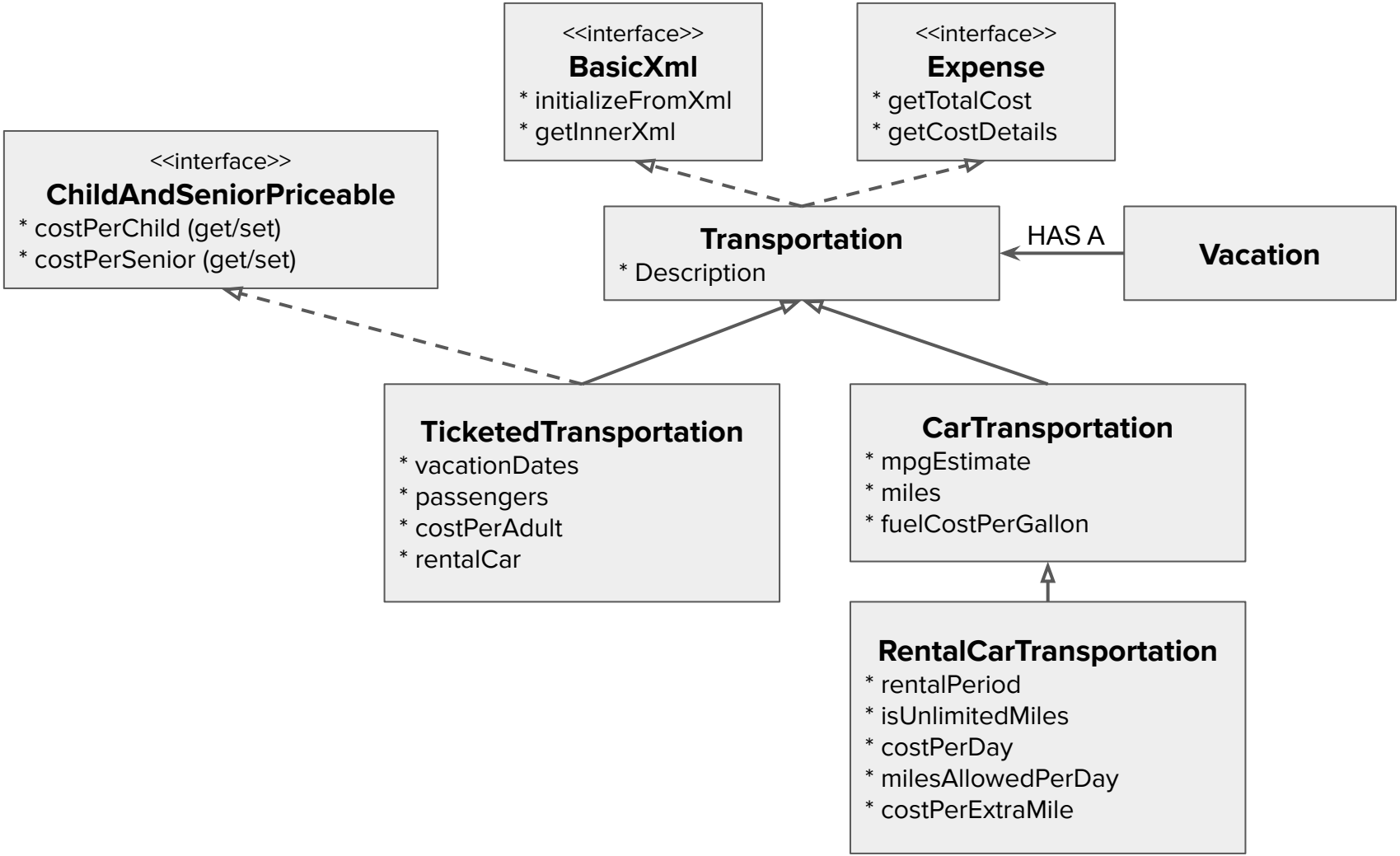


Breakout Rooms

- Pull up the Vacation Expense Estimator app (at the path `<your-student-code-repo>/java/module-1/sample-application`)
- Find the `com.techelevator.vee.model.transportation` package
- Work together to...
  - draw an inheritance diagram for the classes in the package (look for both abstract classes and interfaces)
  - find a composition (HAS-A) relationship where another class contains a reference to a class in the transportation package
  - HINT: start with the `Transportation.java` class

If you have extra time:

- add properties to your class diagram


<<interface>>
BasicXml
* initializeFromXml
* getInnerXml
<<interface>>
Expense
* getTotalCost
* getCostDetails
<<interface>>
ChildAndSeniorPriceable
* costPerChild (get/set)
* costPerSenior (get/set)
Transportation
* Description
HAS A
Vacation
TicketedTransportation
* vacationDates
* passengers
* costPerAdult
* rentalCar
CarTransportation
* mpgEstimate
* miles
* fuelCostPerGallon
RentalCarTransportation
* rentalPeriod
* isUnlimitedMiles
* costPerDay
* milesAllowedPerDay
* costPerExtraMile

Java Docs Example

- All classes implicitly inherit from java.lang.Object
- FileWriter example: https://docs.oracle.com/javase/8/docs/api/java/io/FileWriter.html

Overriding Methods

- toString demo example (in tutorial code)
- Overriding vs. overloading methods (in VEE app)

Polymorphism

In object-oriented programming, polymorphism is the idea that something can be assigned a different meaning or usage based on its context. This specifically allows variables and objects to take on more than one form.

Polymorphism allows you to “swap out” a different implementation at runtime.

Polymorphism

- Examples:
  - HDMI connector
  - List -> ArrayList, LinkedList
  - Map -> HashMap, TreeMap
- Why do we care?
  - Reduces code duplication = more maintainable code
  - De-couples your code
  - Isolates change

Polymorphism

Code examples

Pig dice game

Die

* result : int

---

roll() : int

Player

* name : String
* totalPoints : int
* turnPoints : int

// If roll is 1, bust
// If player holds, add turn points to total and zero out turn points
endTurn()

Game

// Store first player
* players : List<Player>
* activePlayer : Player

---

passTurn() // manage which player is active
endGame()

GameCLI

*

---

promptForUserNameOrder(playerName, order)
promptForFirstPlayer()
kickOffTurn()
promptForTurnDecision()