Week 5

Monday, September 25, 2023 9:18 AM

If you never have to use one of the required 8 methods of a container class that you make, then choose the easiest one to do.

Card

String suit; Heart, Diamond, spade, club
int value; A 2–10 J Q K 1 2 11 12 13

```
Public char getValue(){
  if(value==1){
    return 'A';
  elif(value==11){
```

Deck

ArrayList<Card> deck;

```
Public boolean addCard(String suit, int value){
  deck.add(suit, value)
}
```

Coupling: When objects are dependent on each other.
Loose coupling means that object doesn't need to understand a different object to use it.
Tight Coupling means that an object is highly dependent on a different object.

- Use-a/association - Temporary association with another object.
- has-a/aggregation
- has-a/composition - One of the member variables is an object from another class.
  - Stronger coupling than other has-a
- is-a - inheritance

Overload: To write a second method with the same name as the first, with a different signature.

OOP rules

Class should represent a single abstraction

Class Should Represent a single abstraction
- Should represent a single thing
Data is private
Interface: All of the headers for the methods the user can use.
- cohesive, make the names understandable
- complete, create all methods the class should use
- convenience, make it easy for the user to use
- consistency, keep the same style

Reduce Coupling
Always provide a default constructor

Design Patterns
Singleton Pattern - In the program, only one instance of the class that uses it should be created.
This is because we only want one database to hold data which all programs share.

Syntax: private static className uniqueInstance;
constructors are private (NO PUBLIC CONSTRUCTORS)
create a new method;

```
Public static Daycare getInstance() {
    If (uniqueInstance == null) {
        uniqueInstance = new Daycare();
        Return uniqueInstance;
    }
}
Public static Daycare getInstance(int size) {
    If (uniqueInstance == null) {
        uniqueInstance = new Daycare(size);
        Return uniqueInstance;
    }
}
```

default constructor

nth constructor

To access the class as an object:

```
Daycare daycare = Daycare.getInstance();
```

Day 3

Validating all input: Defensive Programming

Singleton Pattern
- one-time instantiation
- Private constructors
- Public interface to create object but cannot be a constructor
- concurrent access

Iterator - Allows user to move through containers easier
- Establish a starting position
- Peek ability
- Move ? return a value

```
_ = new Scan...
while(_.hasNext()){   Convenience ☆
  X = _.next();       complete
}
_ = new Scan...
while(_.hasNext()){   Convenience
  X = _.next();       complete ☆
  moreOfIt();
}
```

Composition relationship: When a class owns another object (meaning it uses that object as a variable) and is the stronger class, meaning that it uses it.

Aggregation relationship: When a class is used by another object and is the weaker class, meaning that it uses it.

Mapping memory for uniqueInstance

Daycare d = Daycare.getInstance();