CS 1181
Week Two

Reese Hatfield
Clarissa Milligan

Review

- Behavior Modularity
  - Abstract Classes
  - Interfaces
- Separation of Implementation
  - Definitions
  - Implementations

Review

- How can we use this to solve actual problems?
- Data Modeling
- Let’s do an example!

Interface vs. Abstract Class

- Suppose you are creating a media app that allows users to listen to music but also view artwork
- I want to create a class called Media
- Should this be an interface, abstract class, or concrete class?

Media Example

- Considering some of the media items cannot be listened to, what interfaces might make sense to create?

Interface vs. Abstract Class

- Suppose I am creating a system to manage both autonomous and driveable vehicles

Vehicle Tracking System

- Should the following be implemented via an interface, abstract, or concrete class?
  - Vehicle
  - Car
  - UAV
  - Driveable

Practice Problem

- Local Library
- Inventory System
- Managing a large amount of books

Practice Problem

- All books have
  - A Dewey Decimal Number
  - A title
  - A number of days left on loan

Practice Problem

- All Books cost money to borrow
  - Except fiction books are free if you are under the age of 12
- Non-fiction books can have their loans renewed

Practice Problem

- Book Types (DD number, title)
  - Fiction (Cost money)
  - Non-Fiction (Cost Money, return date)

Data Modeling

- Good start to solving *any* problem

- Model how you want your data first
- Implement later
- Adjust model
- Repeat

Data Modeling

- Using the tools we have so far

- How should we model this problem?
- Consider what has “default behavior”

Problem Overview

- All books have:
  - A Dewey Decimal Number
  - A title
  - A number of days left on loan
- Fiction books are free under 12
- Non-fiction books can be renewed

Modeling with Interfaces

- “able” interfaces
- Renewable Interface
- Chargeable Interface
- Abstract Book Class

- Let’s do it!

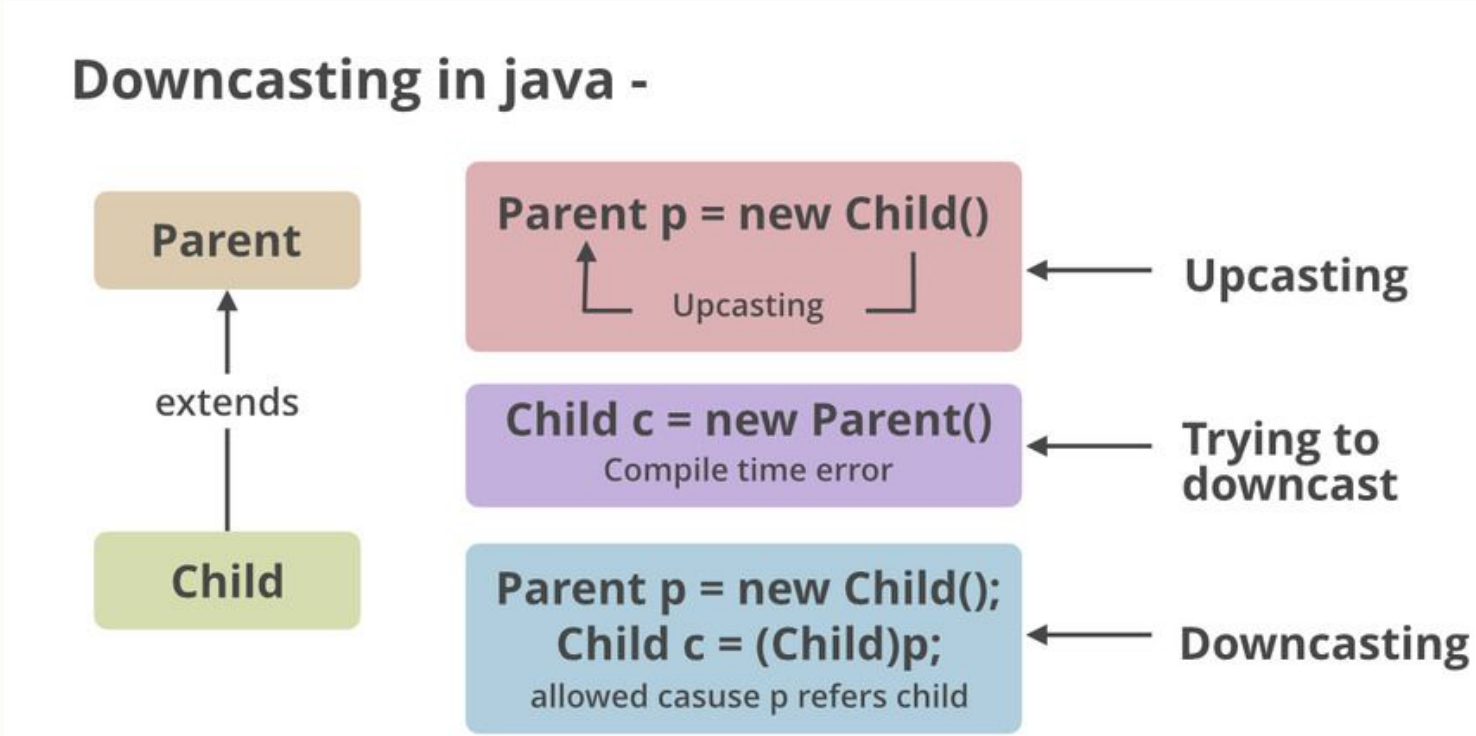
Casting

- Java will let you convert between types
- Cast to interfaces

- checkOut((Borrowable) b3);

- Upcasting vs. downcasting

Casting


Downcasting in java -
Parent
extends
Child
Parent p = new Child()
Upcasting
Upcasting
Child c = new Parent()
Compile time error
Trying to downcast
Parent p = new Child();
Child c = (Child)p;
allowed casuse p refers child
Downcasting

Casting

```
Book b1 = new FictionBook(
    14.01,
    "Twilight"
);
```

Can I cast b1 to a NonFictionBook?

Uh oh!

- ClassCastException

- Occurs when we try to cast to a subclass that our object is not an instance of
- Let’s look at the documentation

Instantiation

- How can I avoid these exceptions?
- In other words, how can I verify the instance of an object?

- instanceof keyword!

instanceof

- Used to verify instantiation of an object
- All lowercase, all one word
- Often used in conditionals

- if (b1 instanceof FictionBook)

What are Exceptions?

- Not a magical entity
- Exception is a standard Java class
- extends Throwable
- What is an Error?
- Let’s look at the documentation

What happens if I extend Exception?

- We can create our own exceptions
- These can be thrown and caught just like any other exception
- What does it mean to be thrown?

Throw keyword

- Used to generate an exception at the current point
- Will cause the program to crash if not caught at a different point

Throw vs Throws

- Throws says an exception *could* occur
- Used for checked exceptions

- Throw creates a new exception at the current point
- Forces an exception to occur