Objectives

- Collections
- Generics
- Eclipse

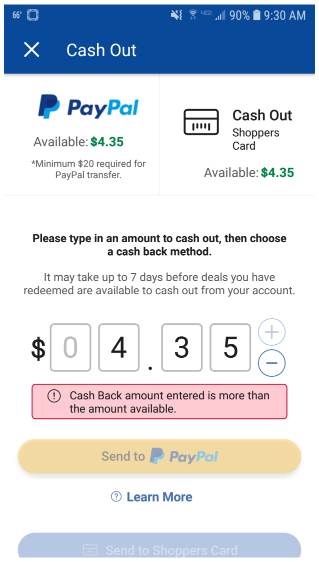
1

Iteration over Code: Assignment 6

- Demonstrates typical design/implementation process
  - Start with original code design
    - Inheritance from GamePiece class
  - Realize it could be designed better
    - Make GamePiece class abstract
    - Use an array of GamePiece objects
    - Easier to add new functionality to Game
- Major part of problem-solving is figuring out how to break problem into smaller pieces
- Reminders
  - Heed my warnings
  - Start simple, small

2

Kroger App Bug


Cash Out

PayPal
Available: $4.35
*Minimum $20 required for PayPal transfer.

Cash Out Shoppers Card
Available: $4.35

Please type in an amount to cash out, then choose a cash back method.

It may take up to 7 days before deals you have redeemed are available to cash out from your account.

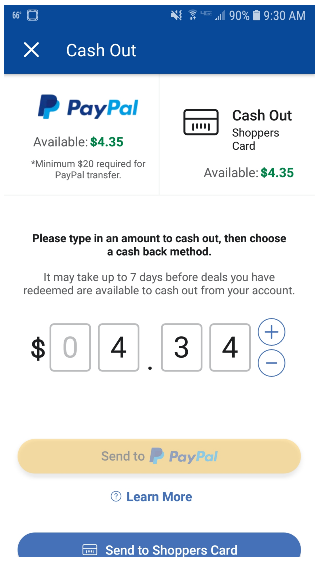
$ 0 4 . 3 5

Cash Back amount entered is more than the amount available.

Send to PayPal

Learn More

Send to Shoppers Card


Cash Out

PayPal
Available: $4.35
*Minimum $20 required for PayPal transfer.

Cash Out Shoppers Card
Available: $4.35

Please type in an amount to cash out, then choose a cash back method.

It may take up to 7 days before deals you have redeemed are available to cash out from your account.

$ 0 4 . 3 4

Send to PayPal

Learn More

Send to Shoppers Card

3

Review

1. What do child classes inherit?
   a) What don't they inherit?
2. How do we specify that a class/method cannot be subclassed/overridden, respectively?
3. Compare and contrast abstract classes and interfaces
4. When should a class be abstract?
   a) If you extend an abstract class, do you have to override all abstract methods?
5. When should you create/use an interface?
6. What is the keyword for specifying that your class adheres to an interface?

4

Using an Interface or Abstract Class

Interfaces

✓ *Any* class can use
  - ✓ Can implement multiple interfaces
- No implementation
- – Implementing methods multiple times
- – Adding a method to interface will break classes that implement

Abstract Classes

- Contain partial implementation
- – Can't extend/subclass multiple classes
- ✓ Add non-abstract methods without breaking subclasses

One Option: Create Both!

- Define interface, e.g., `MyInterface`
- Define abstract class, e.g., `AbstractMyInterface`
  - Implements interface
  - Provides implementation for some methods

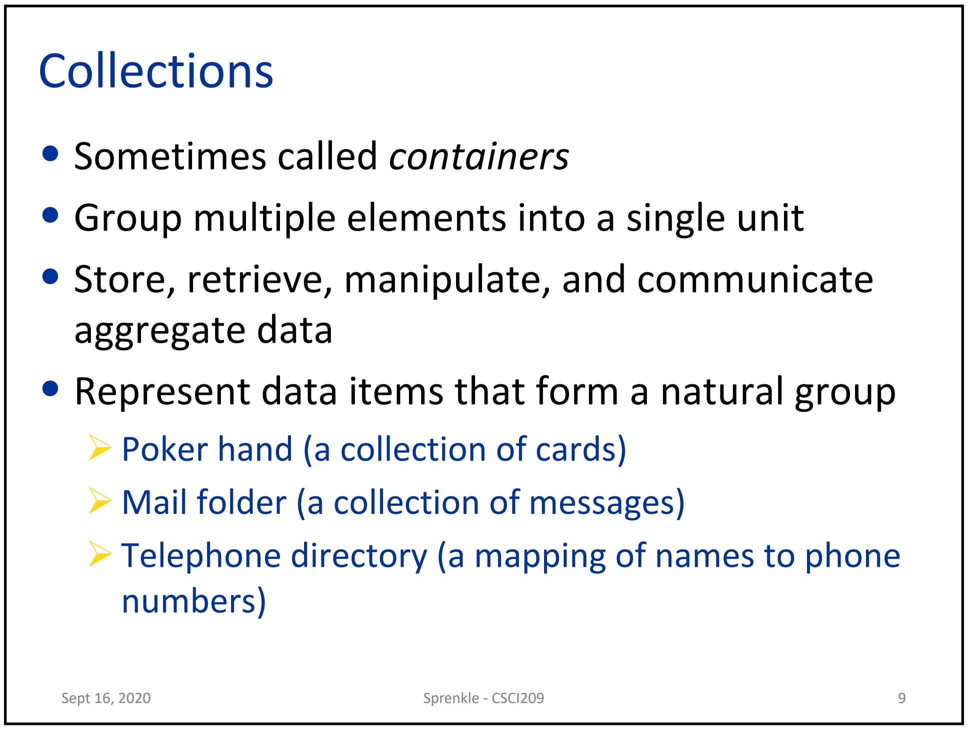
Abstract Classes and Interfaces

- Important structures in Java
  - Make code easier to change
- Will return to/apply these ideas throughout the course
- Concepts are used in many languages besides Java
  - Java enforces the constructs

7

COLLECTIONS

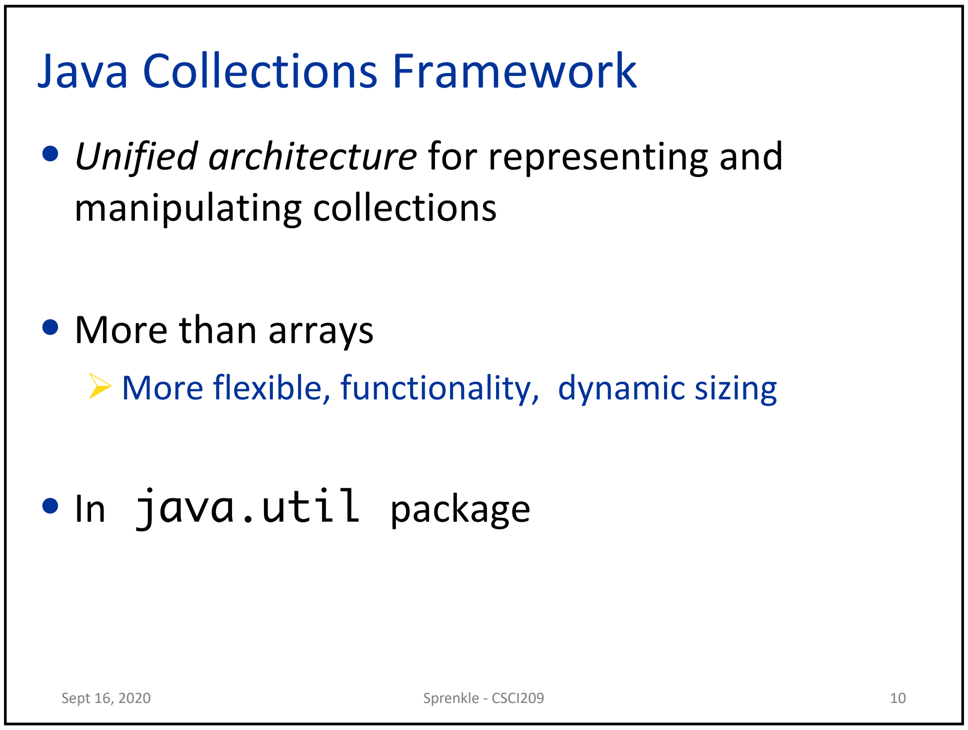
8



Collections

- Sometimes called *containers*
- Group multiple elements into a single unit
- Store, retrieve, manipulate, and communicate aggregate data
- Represent data items that form a natural group
  - Poker hand (a collection of cards)
  - Mail folder (a collection of messages)
  - Telephone directory (a mapping of names to phone numbers)

9



Java Collections Framework

- *Unified architecture* for representing and manipulating collections
- More than arrays
  - More flexible, functionality, dynamic sizing
- In `java.util` package

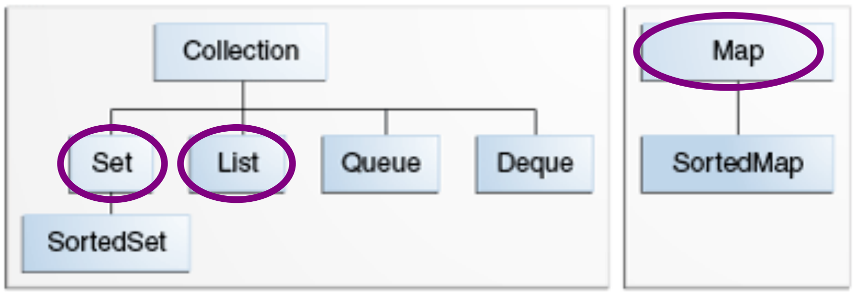
10

Collections Framework

- **Interfaces**
  - Abstract data types that represent collections
  - Collections can be manipulated *independently* of implementation
- **Implementations**
  - Concrete implementations of collection interfaces
  - Reusable data structures
- **Algorithms**
  - Methods that perform useful computations on collections, e.g., searching and sorting
  - Reusable functionality
  - ***Polymorphic***: same method can be used on many different implementations of collection interface

Core Collection Interfaces

- Encapsulate different types of collections



GENERICS

13

Generic Collection Interfaces

- Added to Java in version 1.5
- Declaration of the `Collection` interface:

```
public interface Collection<E> …
```

Type parameter

  - <E> means interface is generic for **e**lement class
- When declare a `Collection`, **specify type** of object it contains
  - Make sure put in, get out appropriate type
  - Allows compiler to verify that object's *type* is correct
    - Reduces errors at runtime
- Example, a hand of cards:

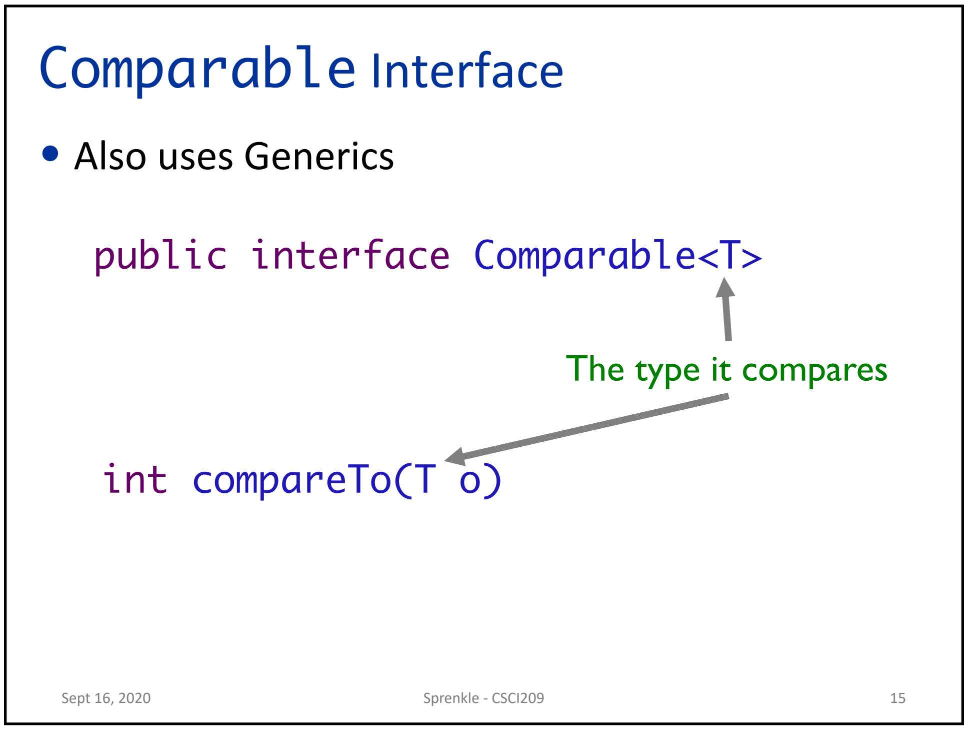
Always declare type

```
List<Card> hand = new ArrayList<Card>();
```

Added in Java 7:

```
List<Card> hand = new ArrayList<>();
```

14



Comparable Interface

- Also uses Generics

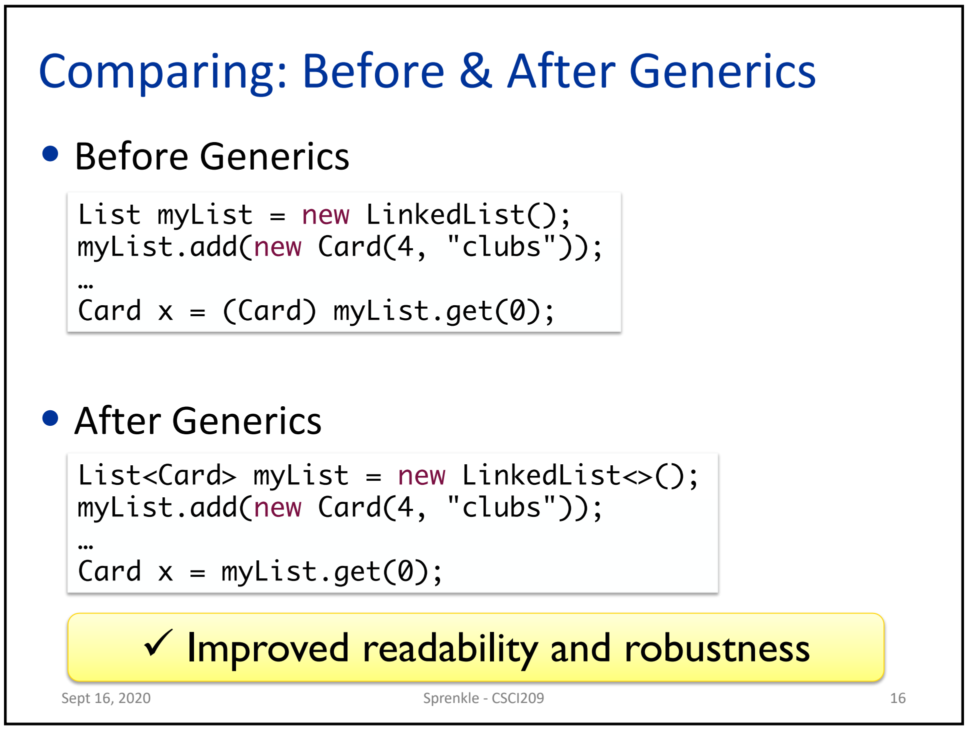
```
public interface Comparable<T>
```

The type it compares

```
int compareTo(T o)
```

Sept 16, 2020 Sprenkle - CSCI209 15

15



Comparing: Before & After Generics

- Before Generics

```
List myList = new LinkedList();
myList.add(new Card(4, "clubs"));
…
Card x = (Card) myList.get(0);
```

- After Generics

```
List<Card> myList = new LinkedList<>();
myList.add(new Card(4, "clubs"));
…
Card x = myList.get(0);
```

✓ Improved readability and robustness

Sept 16, 2020 Sprenkle - CSCI209 16

16

Chicken Comparison

```
public int compareTo(Object otherObject) {
        if( ! (otherObject instanceof Chicken) ) {
            return 1;
        }
        Chicken other = (Chicken) otherObject;
        if (height < other.getHeight() )
            return -1;
        if (height > other.getHeight())
            return 1;
        return 0;
}
```

```
public int compareTo(Chicken other) {
        if (height < other.getHeight() )
            return -1;
        if (height > other.getHeight())
            return 1;
        return 0;
}
```

Types Allowed with Generics

- Can only contain `Objects`, not primitive types
- Autoboxing and Autounboxing to the rescue!

WRAPPER CLASSES

19

Wrapper Classes

- **Wrapper class** for each primitive type
- Sometimes need an instance of an Object
    - To store in Lists and other Collections
- Include functionality of parsing their respective data types

```
int x = 10;
Integer y = Integer.valueOf(10);
Integer z = Integer.valueOf("10");
```

20

Wrapper Classes

- ***Autoboxing*** – automatically create a wrapper object

```
// implicitly 11 converted to Integer,
// e.g., Integer.valueOf(11)
Integer y = 11;
```

- ***Autounboxing*** – automatically extract a primitive type

```
Integer x = Integer.valueOf(11);
int y = x.intValue();
int z = x; // implicitly, x is x.intValue();
```

Convert right side to whatever is needed on the left

Effective Java: Unnecessary Autoboxing

```
Long sum = 0L;
for (long i=0; i < Integer.MAX_VALUE; i++) {
     sum += i;
}
System.out.println(sum);
```

- Can you find the inefficiency from object creation?
- How can you fix the inefficiency?

Autobox.java

Effective Java: Unnecessary Autoboxing

```
Long sum = 0L;
for (long i=0; i < Integer.MAX_VALUE; i++) {
     sum += i;
}
System.out.println(sum);
```

Constructs 2^{31} `Long` instances

- How can you fix the inefficiency?

`Autobox.java`
`AutoboxFixed.java`

Effective Java: Unnecessary Autoboxing

```
Long sum = 0L;
for (long i=0; i < Integer.MAX_VALUE; i++) {
     sum += i;
}
System.out.println(sum);
```

Constructs 2^{31} `Long` instances

Lessons:
- Prefer primitives to boxed primitives
- Watch for unintentional autoboxing

`Autobox.java`
`AutoboxFixed.java`

eclipse

25

2011 Software System Award

The Software System Award is given to an institution or individuals recognized for developing software systems that have had a lasting influence, reflected in contributions to concepts and/or commercial acceptance.

created by IBM.
Eclipse changed the way builders think about tools by defining a set of user interaction paradigms for which domain-specific variants are plugged in and customized for their tool.
Conceived to address perceived shortcomings in proprietary software development tools, Eclipse allowed developers to seamlessly integrate their own extensions, specializations, and personalizations. …

26

2011 Software System Award

It revolutionized the notion of an Integrated Development Environment (IDE) by identifying the conceptual kernel underlying any IDE.

Eclipse was designed as an open, extensible platform for application development tools with a Java IDE built on top. In 2004 Eclipse became a not-for-profit corporation.

The IBM Eclipse team included John Wiegand, Dave Thomson, Gregory Adams, Philippe Mulet, Julian Jones, John Duimovich, Kevin Haaland, Stephen Northover (now with Oracle), and Erich Gamma (now with Microsoft).

27

eclipse https://www.eclipse.org/

- Open source integrated development environment (IDE) for Java
- Has market share for Java IDEs
- Described as “an open extensible IDE for anything and nothing in particular”
- Provides a robust Java development environment
- Incorporates popular software development tools like JUnit and git
- Plugins allow extensibility

28

Project/Code Organization

- `workspace` directory contains all projects
  - Located in your home directory, unless you specified otherwise
- Use **projects** to organize your code
- Within a project
  - `src/` directory contains `.java` files
  - `bin/` directory contains `.class` files
    - Often hidden in GUI

Sept 16, 2020 Sprenkle - CSCI209 29

29

Java Made Easier

- Creating class's basic functionality
  - See `Source` and `Refactor` menus
- Gives you a list of methods for an object
  - After you type `object.`
  - Then shows parameters for methods
- Automatically creates template of Javadoc
  - When you type `/**`
- Autocompletion of variables, methods
- Formatting code …
- Shows unused fields/variables
- Shows compiler errors
- …

Sept 16, 2020 Sprenkle - CSCI209 30

30

Eclipse Demo

Why can a Java IDE provide this functionality?

- Create `Birthday` class
  - Override `equals` and `toString` methods
- Create a new class
  - Generate `main` method, Comments
    - Create a String object, see methods used
- Demonstrate refactoring
  - Rename a field
  - Extract a method (month name)
- Run the Birthday Class (main)
  - Command line arguments

Eclipse Hints

- After you have written a method, type

  `/**`

before the method, and then hit enter and the Javadocs template will be automatically generated for you

- Use command-spacebar for possible completions

Eclipse Discussion

- Helpful hints
  - Control-spacebar
  - Format the file
  - Auto-templates for Javadocs

33

Looking Ahead

- Assignment 7 – due Wednesday
  - Multiple components
    - Eclipse practice
    - Collections
    - Generics
    - Packages
    - …
  - Keep building, based on what we're doing in class

34