CSE 687

Object Oriented Design

Nadeem Ghani
nghani@syr.edu
Office: CST 4-232
Office Hours: Friday 10 AM - 12 PM

TAs
Rui Zuo: rzuo02@syr.edu

- Quiz 1 in class Wednesday Sep 4th
- HW 1 due Friday Sep 6th

CSE 687

Extra Credit

CSE 687

Java Basics

- The Java Tutorials
  - Getting Started
  - Learning the Java Language
  - Essential Java Classes
  - Collections
  - Generics

CSE 687

Java Tutorials

- Learning the Java Language
  - What is an object?
    - state + behavior
    - example: desk lamp
    - example: car (start engine, engage gear, pop trunk…)
  - How are objects modeled by code?
    - fields store state
    - methods represent behavior
  - Data encapsulation
    - road bike gears
    - bluetooth speaker volume level

CSE 687

Java Tutorials

- Learning the Java Language
  - What is a class?
    - many objects of the same type
    - example: cookie cutter -> cookies
    - example: rubber stamp -> impression
  - Car example

CSE 687

Java Tutorials

- Learning the Java Language
  - What is inheritance?
    - many related object types
    - car, truck, motorcycle… Vehicle?
  -
  - Vehicle example
    - which class should have a method to popTrunk()?
    - can changeGear() be inherited from Vehicle class?

CSE 687

Java Tutorials

- Learning the Java Language
  - What is an interface?
    - “... objects define their interaction with the outside world through the methods that they expose”
    - an object’s interface is set of its public methods
    - what is the interface of a car? motorcycle?
  - rewrite Vehicle as an interface

CSE 687

Java Tutorials

- Learning the Java Language
  - What is a package?
    - “A package is a namespace that organizes a set of related classes and interfaces.”
  - JDK 20

CSE 687

Java Basics

```
public class Hello {
    public static void main(String[] args) {
        System.out.println(“hello world”);
    }
}
```

- static
  - per class
  - shared between instances
- void
  - returns nothing (as in C)
- String[]
  - array of String objects
- System.out.println
  - walk thru docs to find System; what is out? println is a method

CSE 687

Java Basics

```
public class Hello {
    public static void main(String[] args) {
        System.out.println(“hello world”);
    }
}
```

- String[]
- primitive vs reference
- int vs Integer
- boolean vs Boolean
- boxing vs unboxing

CSE 687

Java Basics (variations)

```
public class Hello {
    public String sayHello() {
        return “hello world”;
    }
    public static void main(String[] args) {
        Hello h1 = new Hello();
        System.out.println(h1.sayHello());
    }
}
```

- Does a class definition have to start with `public`?
- If a class defines a main method, does it have to be `static`?

CSE 687

Java Basics (variations)

```
/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 */
class HelloWorldApp2 {
    public static void main(String[] args) {
        System.out.println("Hello World!); // Display the string.
    }
}
```

CSE 687

Java Basics

- everything is pass by value
- pass a reference by value allows changing target of reference
  - looks like pass by reference

```
public void add (int x, int y) // implement
public void add (Integer x, Integer y) // implement
```

CSE 687

What is an Object?

Defining a new type in C

```
struct Point {
    int x;
    int y;
}
```

Now make a value of the new type

```
Point p = {1, 2}
```

CSE 687

What is an Object?

Defining a new type in Java

```
public class Point {
     Integer x;
     int y;
     public Point(int x, int y) {
          this.x = x;
          this.y = y;
     }
     public void setZ
}
```

And now make a value of the new type

```
Point p = new Point(1, 2);
System.out.println(p.x); // prints 1
```

CSE 687

Constructors

- Constructor name == Class name; no return type
-
- `new` keyword
-
- default no-arg constructor
- providing a constructor
- classes with multiple constructors
- calling constructor from another constructor

CSE 687

What is an Object?

Access part of a struct ~= access fields of an object
Can a struct contain another struct?
Define two objects
(Single) Inheritance
Implement (Multiple) Interfaces

A C struct is just data; java object is data + methods
Invoke method
static vs instance method

Lets make some variants of what we've seen

CSE 687

More Java

- Constructors
  - instantiation vs definition :: cookie vs cookie cutter
  - naming convention
  - return type missing/implicit
  - arg list
  - if a class doesn't define a constructor, no-arg provided by default
    - super()
  - how many constructors should you have? usability!
- declaration vs assignment
- What does a variable point to?

CSE 687

More Java

- Accessing object fields
  - within class
    - simple name: fname
    - this.fname
  - outside class
    - objectReference.fname
- Calling object methods
- static vs instance
  - fields
  - methods

CSE 687

More Java

- Access control
  - public, protected, not specified, private
  - package
- Return from a method
  - execute all code
  - hit return statement
  - exception
-

CSE 687

More Java

- public class Parent
- public class Child extends Parent
-
- Method can return child of return type
  - Parent p = new Parent()
  - Parent p2 = new Child();

- Method return type can be an Interface
-

CSE 687

More Java

- this keyword
  - within a class refers to the current instance
  - often seen in constructors
    - disambiguate params and fields
    - calling other constructors
-

CSE 687

More Java

- access modifiers
  - basic view
  - but there's a lot more !!
  - language dependent
  - think public/private
  - private by default ??
  - examples

Access Levels

Modifier	Class	Package	Subclass	World
`public`	Y	Y	Y	Y
`protected`	Y	Y	Y	N
no modifier	Y	Y	N	N
`private`	Y	N	N	N

CSE 687

More Java

- static keyword
  - related to class (cookie cutter) as opposed to instance (cookie)
  - examples
    - how many instances of class?
    - static method
    - can a constructor use a static field?
    - can an instance method use a static field?
    - can a static method use an instance field?
- final keyword
- static + final == constant

CSE 687

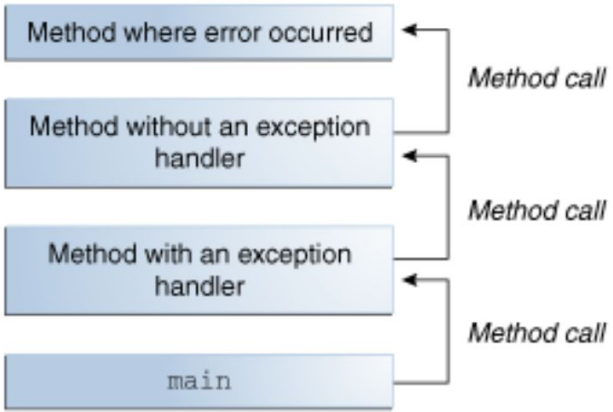
Java Documentation

- api docs
  - module
  - package
  - walk through e.g. java.util.Enumeration
- Java Language Specification (JLS)
  - Intense, not required for this class
  - Worth the effort !!

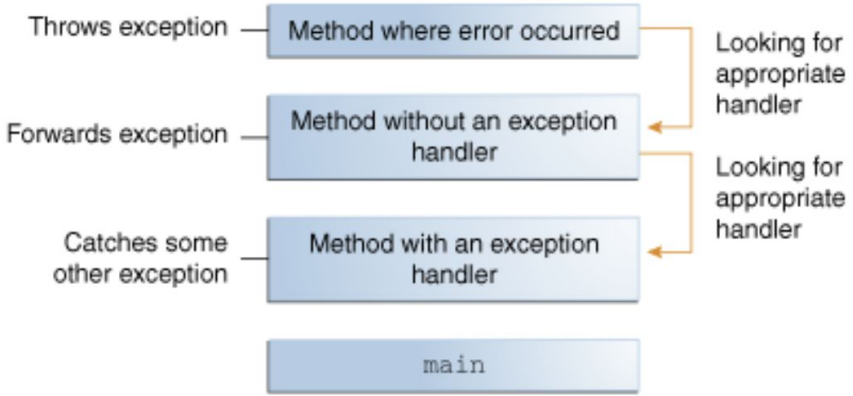
CSE 687

Java Tutorials

- Essential Java Classes
  - Exception (ie Exceptional Event)
  -



The call stack.



Searching the call stack for the exception handler.

CSE 687

Java Tutorials

- Essential Java Classes
  - Exception (ie Exceptional Event)
    - Catch or Specify Requirement
    - Checked Exceptions (java.lang.Exception)
      - expected error, app expected to recover
    - Unchecked Exceptions
      - RuntimeExceptions (java.lang.RuntimeException)
        - application assert failed; best course to quit
      - Errors (java.lang.Error)
        - environment not as expected; best course to quit

CSE 687

Java Tutorials

- Essential Java Classes
  - Exception (ie Exceptional Event)
    - try-catch-finally
    - multiple catch blocks (ordered… like case stmt)
    - catch (Exception | OtherException ex)
    - try (stmt) - catch - finally

CSE 687

Java Tutorials

- Collections
  - Interfaces
  - Implementations
  - Algorithms
  -

CSE 687

Java Tutorials

- Collections
  - Interfaces
    - List
    - Collection
    - Set
    - Map
    - Queue/Deque

CSE 687

Java Tutorials

- Collections
  - Interfaces
    - List
      - Known Implementations
        - ArrayList, CopyOnWriteList, LinkedList…
      - Unmodifiable Lists

CSE 687

Java Tutorials

- Collections
  - Interfaces
    - List

```
boolean add(E element)
void add(int index, E element)
E get (int index)
E remove (int index)
boolean remove(Object o)
int size()
Iterator iterator()
```

CSE 687

Java Tutorials

- Collections
  - Interfaces
    - List

```
List l = new ArrayList();
for (Object o: l) {
    // do work
}
```

CSE 687

Java Tutorials

- Collections
  - Interfaces
    - Set
    - Known Implementations
      - HashSet …
    - Unmodifiable Sets

CSE 687

Java Tutorials

- Collections
  - Interfaces
    - Set

```
boolean add(E element)
boolean contains(E element)
boolean isEmpty()
int size()
```

CSE 687

Java Tutorials

- Generics

```
List list = new ArrayList();
list.add("hello");
list.add(new Integer(2));
String s = (String) list.get(0);
```

CSE 687

Java Tutorials

- Generics

```
List<String> list = new ArrayList<String>();
list.add("hello");
String s = list.get(0);   // no cast
```

CSE 687

Homework 1

1. Write a class, Hello, which has a main method which prints "Hello World" to stdout. This class will be run like so: `java edu.syr.hw1.Hello`.
2. Write a class, Greeting, which has a method greet(), which prints "Hello World" to stdout. This class will be run like so:

```
import edu.syr.hw1.Greeting
public class Runner {
    public static void main(String[] args) {
        Greeting g = new Greeting();
        g.greet();
    }
}
```

CSE 687

Homework 1

3. Write a class, Library, with init() and search() methods.
   The init() method will be passed a String[] as parameter, and it should store these in a field which is an object that implements the List interface.
   In code this would be:

```
List<String> fieldname = new ArrayList<>();
```

   Look in java.util for objects you can use for this. This list is the catalog of the Library, i.e. these are all the publications in the Library available to be borrowed.
   The search() method will be passed a String as parameter, and will return one of the items stored in the Library's catalog.

String book = “Harry Potter…, JK Rowling, WIle..”

CSE 687

Homework 1

4. Complete the implementation of the IntMatrix class.
   The idea is to use a one-dimensional array as a matrix. The field data should be initialized by the constructor to a size large enough to hold all the elements of the matrix. Feel free to add other fields as needed.

   The get() and set() methods should convert the row and column parameters to an appropriate index, and do error checking based on the size of the matrix.

CSE 687

Homework 1

```
public class IntMatrix {
    private int[] data;
    public IntMatrix(int r, int c) {
    //TODO
    throw new UnsupportedOperationException("");
    }
    public int get(int r, int c) {
    //TODO
    throw new UnsupportedOperationException("");
    }

    public void set(int r, int c, int val) {
    //TODO
    throw new UnsupportedOperationException("");
    }
}
```