Introduction to Java

CS9053

Tuesday 6 PM – 8:30 PM

Prof. Dean Christakos

MIDTERM

October 14, 2020

Due: October 20, 2020 11:55 PM

1. Explain the purpose of the keyword ‘this’ in Java

The ‘this’ keyword refers to the current object in a method or constructor.

**this** can also be used to:

Invoke current class constructor

Invoke current class method

Return the current class object

Pass an argument in the method call

Pass an argument in the constructor call

1. Take the Circle type of many examples and assignments in the past. Look at these two methods:

public void modifyArgs(int i, Circle c) {

i = i + 1;

c.setRadius(c.getRadius+1);

}

public void startMethod() {

int i = 5;

Circle circle = new Circle(4);

modifyArgs(i, circle);// output:(6,5)

System.out.println(“the radius of the circle is “ + circle.getRadius());

System.out.println(“i is “ + i);

}

What will the value of circle.getRadius() be and what will the value of i be when startMethod() ends?

Ans: (circle.getRadius():5,i:6)

1. Assume there is a file data.txt that should contain two integers per line (eg, a sample line is “5 10”). This program reads in each line and divides the first number by the second number.

Class Main {

Public static void main(String[] args) {

FileReader file = new FileReader(“data.txt”);

BufferedReader fileInput = new BufferedReader(file);

String inLine = fileInput.readLine();

while (inLine!= null) {

String[] numbers = inLine.split(“ “);

Integer a = Integer.parseInt(numbers[0]);

Integer b = Integer.parseInt(numbers[1]);

Integer c = a/b;

System.out.println(“result = “ + c);

inLine = fileInput.readLine();

}

}

}

1. This is not going to compile because of the IOExceptions raised by the FileReader and BufferedReader constructors. Modify the code so it will compile.

By adding “throws FileNotFoundException,IOException “ after the main function, the problem can be solved.

Class Main {

Public static void main(String[] args) throws FileNotFoundException,IOException{

FileReader file = new FileReader(“data.txt”);

BufferedReader fileInput = new BufferedReader(file);

String inLine = fileInput.readLine();

while (inLine!= null) {

String[] numbers = inLine.split(“ “);

Integer a = Integer.parseInt(numbers[0]);

Integer b = Integer.parseInt(numbers[1]);

Integer c = a/b;

System.out.println(“result = “ + c);

inLine = fileInput.readLine();

}

}

}

1. Add a try/catch block in the while loop to catch any Runtime exceptions. What Runtime exceptions could be raised that you might want to look out for?

Ans: ArithmeticException ,since we can not divide int by zero.

Class Main {

Public static void main(String[] args) throws FileNotFoundException,IOException{

FileReader file = new FileReader(“data.txt”);

BufferedReader fileInput = new BufferedReader(file);

String inLine = fileInput.readLine();

while (inLine!= null) {

try{

String[] numbers = inLine.split(“ “);

Integer a = Integer.parseInt(numbers[0]);

Integer b = Integer.parseInt(numbers[1]);

Integer c = a/b;

System.out.println(“result = “ + c);

}

catch (ArithmeticException e)

{

System.out.println("Oops, you can’t "

+ "divide by zero.");}

finally{

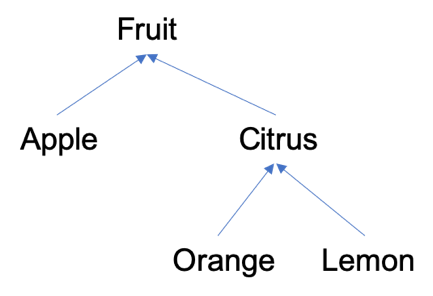
inLine = fileInput.readLine();}

}

}

}

1. Look at this object hierarchy as an example, which we used in a previous assignment:



Let’s say instead we have the following objects:

Aircraft

HotAirBalloon

Glider

Airplane

JetPlane

PropellerPlane

Create a similar inheritance hierarchy for these objects. Explain your reasoning

Aircraft

HotAirBalloon HotGlider Airplane

JetPlane PropellerPlane

Since Aircraft is the superclass. HotGlider and HotAirBalloon is the subclass under aircraft. And jetplane and PropellerPlane is the subclasses under the airplane class.

1. If we wanted to store a bunch of objects in a datatype, you could use an array:

GeometricObject[] geometricObjects = new GeometricObject[10];

or we could use an ArrayList:

ArrayList<GeometricObject> geometricObjects = new ArrayList<GeometricObject>();

Give two reasons you might want to use an ArrayList instead of an array.

1. Arraylist is resizable
2. Arraylist can hold may types of objects
3. In earlier problem sets we had a counter in some objects declared like this:

private static int nextId = 0;

private int objectId;

and in the constructor something like this:

public MyObject() {

this.objectId = nextId;

nextId++;

}

What does the “static” keyword mean in this context and why is nextId declared static?

In Java, static can be used with variables, methods, blocks and nested classes. It is a keyword which is used to share the same variable or method of a given class. Static is used for a constant variable or a method that is same for every instance of a class. And can be used universally.

By using static, the nextid variable can be used in all the instances of the class, so that it can be used in MyObject(), and whenever we created the object, the number will universally increase , not just in this scope.

1. What is the difference between an overridden and overloaded method?

Overloading occurs when two or more methods in one class have the same method name but **different parameters**. Overriding means having two methods with the **same method name and parameters** (i.e., method signature). One of the methods is in the parent class and the other is in the child class.

1. Take that object from an array declared as Object[] objs. The result of (new Object()).toString() is something like java.lang.Object@15db9742 - the object type followed by the @ sign and a unique identifier. In the objs array, let’s say we execute the following:

objs[5] = new Integer(20);

Object myObj = objs[5];

What is the output of myObj.toString() ? Explain why.

Ans: the output is 20. Since the “tostring” method is override by the object, not the whole array.

(new Object()).toString():

This output is, class name, then ‘at’ sign, and at the end hashCode of object. All classes in Java inherit from the Object class.

Override toString() let us get a proper output when an object is used in print() or println().

1. Remember in an earlier lecture when discussing Abstracts, Interfaces, and Generics we had the class Fruit, declared as:

public abstract class Fruit { public Fruit(String name) { /\*etc\*/ } /\* etc \*/ }

We created Apple and Orange objects. Why didn’t we ever simply do the following:

Fruit f = new Fruit(“Apple”);

Because it is an abstract class, it can not be created as an instance.

But abstract class can be used as a data type.

1. In this week’s lecture, we had two abstract classes, Animal and Fruit, from which we made various subclasses. Some of those subclasses implemented the interface Edible. Why did we choose to make classes subclasses of Animal and Fruit, but have Edible as an interface? To put the question more generally, why would you design an object as a subclass of another object vs. why would you create an interface and have an object implement an interface? (There’s no single right answer to this question, but there are definitely wrong answers)

The intent of an interface is to specify common behavior for objects. For example, you can specify that the objects are comparable, edible, cloneable using appropriate interfaces.

But for superclass and subclass, it is about hierarchy, not the concept of implement.

And we cannot inherit multiple class to the subclass but we can implement interfaces.

In this case, Animal and Fruit are in different classes that you cannot cross but they can both implement edible().

**Part II: Coding Section**

1. **Inheritance**

Demonstrate your ability to program using inheritance in the Java programming language.

Demonstrate your knowledge of proper equals implementations in the Java programming language.

Demonstrate your knowledge of variable arguments and enum types in the Java programming language.

### Instructions

* There are three tasks to complete
  + Create an object hierarchy which mimics the classification for players of [winter sports](https://en.wikipedia.org/wiki/Winter_sport).
    - The hierarchy should be contained within package edu.nyu.cs9053.midterm.hierarchy
    - There should be a base class called WinterSportPlayer
    - There should be the following subtypes:
      * Luger
      * IceSkater
      * Skier
      * SpeedSkater
      * Curler
      * Sledder
      * Bobsledder
      * CrossCountrySkier
      * MogulSkier
      * FigureSkater
    - Each of the subtypes should have the following methods:
      * getName returns a name for the player as a String. This value is per object and not defined by the class it should be used for equality.
      * getAge returns the age of the player as an int. This value is per object and not defined by the class it should be used for equality.
    - If appropriate, the subtypes should have the following methods:
      * getSkateSize returns the size of the player's skate as an int.
      * getSledColor returns a String representing the color of the sled the player users.
      * getSkiLength returns the length in centimeters of the player's skis as an int.
    - For each class which has no sub-types itself, add an instance field to the class particular to the type which is also used in the equals methods. For example, a curler might have a field “trouserPattern” or “brushLength,” that other classes don’t have. There should be a field that is unique to that class that no other class has.
  + Provide implementations of the equals and methods for each concrete class within package edu.nyu.cs9053.midterm.hierarchy
    - Note, these methods are testing equality for the objects and so should only include checks on type information and object specific values.
    - Note, do not share code with super classes.

### Implementation

* Ensure your code is correct by compiling and testing it
* A portion of your grade will be based upon readability and organization of your code.
  + Follow the naming guidelines of lecture
  + Break large functions into multiple functions based on logical organizations

1. **File IO and Exceptions**

Create a Java class Matrix to represent bidimensional matrices of real numbers. The class should export the following methods:

• Matrix(int n, int m) : constructor that creates a matrix of size nxm, with all values initially set to 0;

• void save(String filename) : that saves the content of the matrix on the file specified by filename;

• static Matrix read(String filename) : that reads the data about a matrix from the file specified by filename, creates the matrix, and returns it;

• Matrix sum(Matrix m) : that returns the matrix that is the sum of the object and of m, if the two matrices have the same dimensions, and null otherwise;

• Matrix product(Matrix m) : that returns the matrix that is the product of the object and of m, if the two matrices have compatible dimensions, and null otherwise. Review how to multiply two matrices at <https://www.mathsisfun.com/algebra/matrix-multiplying.html>. Hint: implement a method “dotproduct” and use that in your product method

Define the exceptions that are necessary to catch the possible errors that can occur in the class Matrix:

• ExceptionWrongMatrixValues that is thrown in the method read() if the data on the file does not correspond to numeric values, or if the data are not consistent with the form of a matrix (e.g., the rows have different length);

• ExceptionWrongMatrixDimension that is thrown in the method read() if the data on the file do not correspond to the dimension of the matrix. Modify the class Matrix in such a way that it generates the new exceptions when necessary.

I have provided some sample input and output files and code in main(). A matrix file should have the first line specify the dimensions in integers, and the following lines corresponding to and m x n matrix with double values.

A few hints for file reading and processing:

* If you’ve read in a line, inLine, the method inLine.split() will return an array of Strings of the contents of the line separated by spaces. Eg, for the line

5.5 2.3 4.5

inLine.split() will give an array

[“5.5”, “2.3”, “4.5”]

* If you have a string representing a numerical value, to convert it to an Integer, use the method Integer.parseInt(). For a double value, use the method Double.parseDouble()