Exceptions

When do we need exception mechanism?

When and why use exceptions?

The purpose of exceptions

Exceptions

An exception is thrown when there is no reasonable way for the code that discovered the error to know what to do about the error.
For example, if your code tries to open a file with "new FileInputStream(filename)" but the file doesn't exist, then the FileInputStream constructor doesn't know what to do. The fact that the file doesn't exist may be something that you know how to recover from, or it may be a fatal error that should cause your program to stop.
The FileInputStream constructor throws a FileNotFoundException and lets the calling program deal with the problem. If your program knows what to do about a file not found, then you probably surrounded the call to the constructor with a try / catch block, and can recover from the problem using code in the catch block. If the error is totally unexpected and shouldn't happen, then you may let the error get passed up the line until the program itself throws the exception and the Java runtime finally catches the exception and your program is terminated.

How do you know if an exception will be thrown from some code?

Checked exceptions are the exceptions that might result from normal operation of the program. All subclasses of Exception (except for RunTimeExceptions) are checked exceptions. The compiler "checks" that your code deals with every checked exception that might occur.
Every method that throws a checked exception must say so in the method header. So you can look at the code itself and see that it says it will throw an exception. Any method that throws an exception should document that possibility in the javadoc comments and so you should be able to look at the documentation and see that a method might throw an exception.
Unchecked exceptions are a little harder to detect. Unchecked exceptions generally result from a bug in your code, and so they should never happen. One very common unchecked exception is NullPointerException, which is thrown when an application attempts to use null in a case where an object is required. These cases include (but are not limited to):
Calling the instance method of a null object.
Accessing or modifying the field of a null object.
Taking the length of null as if it were an array.
Accessing or modifying the slots of null as if it were an array.
Throwing null as if it were a Throwable value.

What is the difference between throwing an exception in the beginning of a method and using try-catch inside of a method?

In the method header, you can specify that you *might* throw a checked exception. For example, if you write a method myMethod that calls "new FileInputStream(filename)" then you might get a FileNotFoundException. If you do not catch this exception in your method, then myMethod

must say in its header "throws FileNotFoundException". This tells the compiler that myMethod might cause this error, but that myMethod is not going to do anything about it. Therefore any method that calls myMethod should be prepared for this to happen, either by catching the exception or by passing it up the line to its caller.
See next section for try-catch.

try, catch, finally

Execution enters a try{} block
If an exception is thrown within the block, then execution jumps immediately to the appropriate catch{} block. (If no appropriate catch block is found, then control passes to the caller of the method. The system looks for an appropriate catch block there, and so on up the line.)
If an exception is not thrown, execution proceeds all the way through the try{} block, then skips all the catch {} blocks
In either case (exception or not), when execution exits the try{} block, it proceeds to a finally{} block if there is one.

Throwing and catching an exception

Under some circumstances you might want to throw an exception that you have defined. However, most of the time you will just catch exceptions that happen in the Input/Output system and try to recover from them as best you can.

Exceptions. What do I really need to understand?

You need to know how to surround a potentially risky operation that might throw an exception with a try-catch block. For example, if I tell you how to use method doIntervalCheck() and that "it may throw checked exception TransitRouteException" then you should be able to write a try-catch block that would use doIntervalCheck() and protect against the occurrence of a TransitRouteException. I would tell you what to do in the catch block.

Input and Output

Input / output stuff.

Details of I/O

When do we use input/output stuff?

What is I/O?

The stream, input/output, buffer, (I/O stuff)

The I/O stuff is a little overwhelming. Much of the vocabulary you used and are currently using with respect to files (to and from), performance, etc. are not things that I am familiar with. Overall need more explanation for I/O.

I agree it can be overwhelming at first.

However, there really is a basic concept: a stream is a Java object that moves bytes (little chunks of data) between your program and the outside world.
All the rest of the I/O stuff is designed to let you say exactly where those bytes are coming from or going to (a file, the console, …), how the bytes are interpreted (are they characters, are they numbers, …) and other "administrative information" (what line number, move bytes one-by-one or in a block, …) .
Almost all programs that you will deal with in the near future read and write characters (text) to and from the console or disk files. I gave examples of each of these four cases (read keyboard, write display, read file, write file).

Input/Output – do we have to know all the subclasses? Exactly what do we have to know about them for the final?

Will we be required to write pieces of code using input / output? How could we do this?

What are we expected to do with Input/Output and I/O files?

Input Stream and Output stream. eg, the difference between PrintStream and PrintWriter. To what extent are we supposed to know all about those?

I expect you to understand the basic concepts described above.
You do not need to know all the I/O related subclasses.
You should be able to read code that does simple input and output and make knowledgeable statements about what it is doing.
You should know that there are classes in the java.io package that can be used to construct a stream that is connected to a file for input and output. You should know that the System class provides streams that are already connected to the keyboard and display.
I'll talk about the exam more in the review sheet.

Interfaces, Inheritance, Abstract Classes

What is the difference between inheritance and interface? Both use the keyword "extends" in their implementation. I'm having trouble distinguishing between the two.

The difference between an interface and a class is that an interface defines the behavior (method names) that a class must have, but a class that "implements" that interface actually has code that does something for each method. The interface just has the required method names, the class has the method names and the code to implement the methods.
An interface A defines a set of method names and parameters. It is possible to define an interface B by saying "you must implement all the methods defined in interface A, plus a few special ones that I have defined here in interface B." In that case, interface B "extends" interface A.
A class defines constructors, methods, and variables. Any class can do that from scratch. However, if you want to take advantage of code that has already been implemented that does tasks similar to what your class should do, then you can define the new class and say that it "extends" the base class.

For example, "public class TransitBus extends LocatedVehicle". In this case, all the methods of a LocatedVehicle are already implemented, and so the TransitBus class does not have to re-implement the various methods associated with location, it just inherits them from LocatedVehicle. However, the LocatedVehicle class doesn't know anything about route numbers, so TransitBus implements that little extra bit of code.

"extends" and "super" and "implements"

See previous paragraphs for "extends" and "implements".
"super" refers to the class that this class extends. Using the TransitBus example, the TransitBus toString() method can refer to the LocatedVehicle toString() method by saying "super.toString()". Similarly, the constructor for TransitBus uses the LocatedVehicle constructor with the line "super(vin,lat,lon);"

I don't know how / when to use abstract classes.

abstract classes – go over again.

interface versus abstract class. When do you use them? Which is best to use?

Generally speaking, the design pattern is to use both when you are defining a general-purpose capability that you expect to implement several times or in several different ways.
First, you specify the behavior you want using an interface. This gives a clean definition of the behavior, and makes it possible for any designer to implement the methods in any class.
Second, you write an abstract class that implements most or all of the methods in a general-purpose way. The designer who only needs one little detail changed can extend the abstract class and implement just the changes. The developer who needs a very special implementation can ignore the abstract class and start from a clean sheet of paper and implement all the methods in the interface.
Any class can implement an interface. If it actually provides code for every required method in the interface, then it can be a "concrete" class and it can be used to create new objects. However, if it only implements some of the required methods in the interface, then it must be declared abstract and its subclasses must implement the remaining methods.

Arrays

Array Syntax

Variables that hold references to arrays are declared like any other variable. The type of the variable is shown as <the type it is holding>[]. So you can say:
int[] myIntArray = new int[10];
Dog[] myDogArray = new Dog[23];
The numbers shown above (10, 23) are the sizes of the arrays. The elements in an array with ten elements are accessed using 0..9 as the index values.
The elements in the array are accessed using square brackets around the index value. To set, then retrieve the first element in the int array declared above, you would say:
myIntArray[0] = 16;
int val = myIntArray[0];

Note that the values in the array are initialized to zero or null when the array is created, it is up to your code to fill in meaningful values before you retrieve them.

Arrays, [something …], arrays vs ArrayLists

In the API documentation, there are Arrays and Array – what is the difference? Which one do we use? Capital letter "a"?

The Arrays class (big "A") in the java.util package contains numerous static methods for dealing with arrays. This class is just a convenient place to collect a bunch of methods that are used to manipulate arrays. You can use the methods in this class when they do something that you need. You can create and use arrays (little "a") whenever you want a nicely ordered group of elements all of the same type. Sometimes an array is simpler to use than an ArrayList, just because the notation is simpler. However, arrays are more limited in what can be done with them, and so an ArrayList is often a flexible alternative to consider.
There is an interface Array in package java.sql. Ignore it unless you are doing SQL database programming.
There is a class Array in packjage java.lang.reflect. Ignore it, it is for internal use by Java.

Reading the documentation

How to use the methods defined in the various Java doc files. Is there another source or sources to consult that would explain how to implement some of the methods not fully understood based on the Javadoc files?

The Java tutorial often has good information about the important classes and their methods.
A good programmer's reference (as opposed to a school textbook) will often have useful information. I use Core Java by Horstmann and Cornell.
Sun has a developer services website that has lots of useful faqs and documentation.
http://developer.java.sun.com/developer/index.jshtml

I find it difficult to get useful information out of the documentation libraries. Specifically, the parameter requirements are often confusing, requiring classes that I'm not sure how or where to instantiate in my code.

For the documentation of the homework skeletons, I've tried to be careful to document what you need to pass in as parameters. If that's not clear, try going back over the description of the homework project. Also, look at the code that has been supplied.
For the documentation of the Sun libraries, it may be that you are unfamiliar with the classes that are being requested as parameters. (Not surprisingly, since it's all new!) I have the same problem when I'm exploring a new area, and the best I can suggest is to be patient and follow each of the class links and see what the classes are that are being passed around.

Static methods and main

Static methods

How to set up a main method and how it works.

main method (String[ ] arg) – I don't get it.

Static methods and variables exist before any objects have been created. At the very least, you need a static main method so that java knows where to start your program. Generally the main method creates a few objects, then turns control over to them for the rest of the program run. See lecture 22, August 19, for more information.

Development tools and techniques

How to create batch files

Use any text editor (notepad.exe, wordpad.exe, etc) and type in the commands that you want to run. Take a look at some of the bat files I have provided you. Right-click on one of the files, then select "edit". It will open the file in notepad.exe (probably) and you can see what's there.

I'm still confused on the difference and relationship between different development tools, editors, compiler, Java virtual machine

Some sort of text editor is used to create the source files (eg, Dog.java) that contain statements in the Java language.
There is an editor that is part of BlueJ. jEdit is an editor. Any text editor like notepad or wordpad can be used, although they don't work as well as an editor designed specifically for editing code.
The compiler (javac.exe) translates your code from source form (Dog.java) to binary form (Dog.class). BlueJ and jEdit both know that you will want to do this, and so they supply buttons or commands that you can use to start the compiler running.
The java virtual machine (java.exe) reads the binary class files produced by the compiler and does whatever they specify. BlueJ knows that you will want to do this, and so it starts a JVM running for you. With real application programs, you start the JVM running yourself with the java command. For example, in order to run the Employee program in ex142\lect22, at a Windows command prompt you could type ">java -classpath . Employee".

How to start designing a project? What are the considerations? Every time I start the project homework, the most troublesome thing is trying to figure out what happens between classes and what are the parameters passed between methods. It makes me focus on filling out the "missing" classes/methods, instead of understanding how the whole project works.

Designing an entire project is similar to designing a single class in that you need to think about what the basic data structures are and understand the flow of data throughout the program.
In order to do interesting things like graphics or real-time displays, it is usually necessary to use existing class libraries to do the complicated parts of the task.

In any real-world application, there will be aspects of the design that use existing classes, certainly for input and output, and probably for other aspects of the program too. You probably will not have a complete understanding of the inner workings of all these other classes, but you will be able to use them because the methods that you need are well defined.
All of the example programs that I supplied in ex142 during the quarter are complete programs specified by one or just a few files. Read through those if you want to explore complete programs that do not refer to a large base of existing classes.

Specific Java language issues

Casting of different types of objects

When you specify a cast from one type to another in a Java statement, you are telling the compiler that it should treat the object as being of the new type. This is a promise by you that the new type is appropriate for the object.
We have used casts when we retrieve objects from an ArrayList. The ArrayList get(idx) method returns objects as being of class Object, however we often know more about them than that. So we can tell the compiler "I know this object is really some sort of Shape, so let's say that it's a Shape and let me use all the Shape methods on it."
Shape s = (Shape)myShapes.get(idx);
If your promise to the compiler turns out not to be true, then an exception is thrown when your program runs and gets to this line of code.
You can also cast primitive types. This is necessary when you are doing something that may not work well, and you need to tell the compiler something like "yes, I know that there may be a loss of precision, but I want to do it anyway."
double x = Math.PI;
int pie = (int)x;

How to create a package

Include a package statement at the front of each java source file in the package. For example "package hw7;"
Put all the classes in the package in a subdirectory named the same as the package. For example all the classes in package hw7 are in directory hw7.

c.getClass().getName() - when you have something like this, is it implemented from left to right?

Yes, it is evaluated left to right. "c" is a reference to an object. getClass() is an Object method. It returns an object of type Class describing the class of "c". getName() is a Class method. It returns a String containing the name of the class.

Making new objects from inside another

see next item

Referencing classes not inside the file we are currently working with.

You create an object of some other class by using the constructor of that class.
Dog rover = new Dog("Jake");

Calling methods and constructors from other classes

Once you have a reference to an object, either by creating the new object or because a reference was passed in to your method as a parameter, then you can use the methods defined for objects of that class.
rover.sleep();

(String[ ] arg) parameter variables

declaration of arrays inside parameters vs outside

The information in a parameter list for a method tells you what must be passed in to the method. So, for example, doCommand(String[] arg) says that anybody who calls the doCommand method must supply a String array object as the only argument. The statement "String[] arg" does not allocate an array anywhere, it just says that when this method is called, the caller must supply a reference to an array.
In order to actually allocate an array object, you need a statement like the following somewhere in the calling method or elsewhere.
String[] usr = new String[10];

break and continue statements

break takes you out of a loop. It lets you jump to the end of a for loop or while loop and exit the loop statement immediately.
continue takes you to the end of a loop and starts the next iteration. It lets you skip whatever code is remaining in this iteration of the loop and go on to the next iteration.
See lecture from July 22.

which class can use another's methods; it appears to be method sharing or something. Still a bit fuzzy.

By "use another's methods" I'm not sure if you mean "call the methods of another class" or "use the implementation of methods by another class to be my own implementation."
"Call": If you have a reference to another object, you can call the public methods of that object. If your class is in the same package as the other class or is a subclass, you can also call the package and protected methods of the other class.
"Reuse implementation" If you want to use the implementation of a method from some other class in your own class, you need to "extend" the other class. This makes your class a subclass of the other class. For example, TransitBus is a subclass of LocatedVehicle, and so TransitBus can reuse the implementation of getLocation() from LocatedVehicle.

I can generally read Java file, but it's really hard to write "perfect" Java.

Practice.

How to write the code. The syntax.

Practice. Refer to the examples and read through them to understand what each part is for.

In Hw 6 – doRoute

The idea of doRoute is that it sorts the list of buses according to route number, then it goes through and counts how many buses are on each route. One way to do this is with nested loops,

with the inner loop counting how many buses are on a particular route, and the outer loop iterating until we have run out of routes.

TreeSet "sorted in ascending element order" So TreeSet also has order? Is this feature same as List?

The TreeSet stores its elements in order, but they cannot be accessed by index like you can with a List. You still have to use the Set methods for access. You can use an Iterator to look at all the elements, and the iterator will return them in sorted order. This is the difference between a regular Set and a TreeSet.

How to get information eg, (data about Bus running in the system) from online.

The real-time information about buses is made available over the network by a computer at the UW. In homework 7 I supplied a program called BusRcv that can read that data and make it available to your program. Make sure your computer is connected to the network (via modem or whatever), then run BusRcv with the batch file runBusRcvNet.bat. Double click on the batch file to run it. Once that program is running, use runBusDisplayRcv.bat to start your program running.

command flow. HW 5, q1, give new example.

Assume that we have a TransitBus object named myBus.
Then System.out.println(myBus) will call the toString() method of TransitBus.
Consider the toString() method in TransitBus.

```
public String toString() {
        return super.toString() + " on route "+route;
}
```

The first thing that method does is call the toString method of its superclass LocatedVehicle, which is:

```
public String toString() {
        return super.toString()+" at "+location.toString();
}
```

The first thing that method does is call the toString method of its superclass Vehicle, which is:

```
public String toString() {
        return this.getClass().getName()+" "+vin;
}
```

This method uses this.getClass().getName() to get the name of the class of the object that we are trying to print out, then adds the vehicle id number to the name String. That new String is returned to the caller.
The caller (toString in LocatedVehicle) takes that String and adds to it the result of the toString method in the Location class, which is:

```
public String toString() {
        return "("+Double.toString(latitude)+", "+Double.toString(longitude)+")";
}
```

This method calls the Double.toString method twice to put the latitude and longitude in String format, and returns the result.

So toString in LocatedVehicle adds the location string to the vehicle string and returns the combination to its caller, toString in TransitBus. The toString method in TransitBus adds the route number information to the string and returns that.
Finally, control passes to the println() method and the string is printed out.

Nested loops, more examples

see Looper.java in ex142\lect23.

Final Exam

How long will the final exam be?

1 hour.

The recent lecture material such as errors, throws, exceptions etc is unclear. Please specify what is important to know for exam concerning these topics

See the sections above in which I reviewed exceptions.
I will talk about this in the review on Wednesday.

Which sections to be covered in the final exam?

I will talk about this in the review on Wednesday.

CSE 143

Will CSE 143 be going on talking about Java? Or is there any class opened for Java programming?

CSE 143 is taught in Java, and it is the next class you should take if you want to learn more about programming in Java.