Beck 325 midterm study guide

Overloading vs. Overriding

Overloading – Uses same method name but has different parameters

Overriding – Uses same method name with the same parameters

* An overriding method can extend or replace the functionality of the superclass method
* Use reserve word super to call a superclass method.
* Unless specified otherwise, the subclass constructor calls the super class constructor with no arguments.
* To call a superclass constructor, use the ‘super’ reserved word in the first statement of the subclass constructor.
* The constructor of a subclass can pass arguments to a superclass constructor, using ‘super’ word.

Method Access Modifiers

Public – The code is accessible for all classes.

* Accessible to class, package, subclass and world.

Private – The code is only accessible within the declared class.

* Accessible to class.

Protected – The code is accessible in the same package and subclasses.

* Accessible to class, package, and subclass.

Non–access Method modifiers

Final – Attributes and methods cannot be overridden/modified from the outside. Finalizing the implementations of classes, methods and variables.

Static – Attributes and methods belongs to the class, rather than an object. Used for creating class methods and variables.

Side: Creating a ‘static final’ method would declare it a constant.

Objects

* An object has:
  + A State – its descriptive characteristics
  + Behaviors – what I can do (or what can be done to it)
* The behavior of an object might change its state.
* The object is defined by the class.

Subclasses and Superclasses

* First, let’s define what a class is:
* The class is a blueprint of the object
  + It uses methods to define the behaviors of the object
* The class represents the concept.
  + Class = blueprint
* Subclasses and Superclasses
* Definitions:
  + Superclass – a more generalized class (Vehicle)
  + Subclass – a more specialized class (Car)
* Subclasses inherit from the superclass
  + All public methods that it does not override
  + All instance variables
* The subclass can:
  + Add new instance variable
  + Add new methods
  + Change the implementation of inherited methods
* A subclass can override a method of the superclass by providing a new implementation.
* The subclass inherits from the superclass and ‘extends’ the functionality of the superclass.
* A subclass cannot directly access private instance variables of the superclass.
* Overriding the superclass method:
  + Use the same method that is in both the super and subclass, must take all same parameters
* The subclass ‘inherits’ data (variables) and behavior (methods) from the superclass

Polymorphism

* We ask multiple objects to carry out a task, and each object does so in its own way.
* Polymorphism makes programs easily extensible.
* Dynamic method look-up:
  + Method calls are always determined by the type of the actual object, not the type of the variable containing the object reference.
  + Dynamic method look-up allows us to treat objects of different classes in a uniform way.
* A subclass reference can be used when a superclass reference is expected.
* Allows us to manipulate objects that share a set of tasks, even though the tasks are executed in different ways.
  + An abstract method is a method whose implementation is not specified.
  + An abstract class is a class that cannot be instantiated.

Abstract Classes and references

* If it is desirable to FORCE subclasses to override a method of a base class, then you can declare a method as ABSTRACT
* You will not be able to instantiate an object that has abstract methods.
  + However, you can declare an object reference whose type is an abstract class.
  + The actual object to which it refers must be an instance of a concrete subclass.
* CONCRETE class – a class that can be instantiated
* The reason for using abstract classes is to force programmers to create subclasses.

Inheritance and Interfaces (Extends vs. Implements)

* Inheritance – Where one object acquires all the properties and behaviors of a parent class. You create new classes that are built upon existing classes, when we inherit, we use the methods from the existing class and reuse methods/fields of the parent class.
  + Parent-child relation ship
    - Super and subclass
  + A subclass inherits data and behavior from a superclass
  + Can use a subclass object in place of a superclass object
  + A subclass inherits all methods that it does not override
  + A subclass can override a superclass method by providing a new implementation.
* Interface is a special type of declaration that lists a set of methods and their signatures
  + A class that ‘implements’ the interface must implement all of the methods of the interface.
    - All methods in an interface type are abstract (they have a name, parameter, and a return type but they don’t have an implementation)
    - All methods in an interface type are public.
    - An interface type cannot have instance variables.
    - An interface type cannot have static methods.
  + A class can implement more than one interface at a time
  + A class can EXTEND only one class but implement many interfaces.
  + An interface can extend another interface, just as a class can extend another class.
* Multiple inheritance is not allowed, interfaces are not classes and an interface can extend more than one parent interface.
* The extends keyword is used once, and the parent interfaces are declared in a comma-separated list.
* Interface can have only abstract methods. Abstract class can have abstract and non-abstract methods.
* Final Variables: Variables declared in java interface, abstract class may contain non-final variables
* An abstract class can have final, non-final, static and non-static variables, interface can only have static and final variables.
* Abstract class can provide the implementation of interface, interface can’t provide the implementation of abstract class.
* Inheritance vs. Abstraction – A java interface can be implemented using the keyword ‘implements’ and abstract class can be extended using keyword ‘extends’.
* An interface can extend another java interface only, an abstract class can extend another java class and implement multiple java interfaces.
* Members of a java interface are public, java abstract classes can have class members like private, protected, etc.

CompareTo Method

* Comparable Interface
  + Requires only the CompareTo Method
  + Used in comparing two objects
  + It is called on by one object and is passed another ex. a.compareTo(b);
  + Uses a special type of parameter that allows it to work with any type, <T> type is used as a place holder for an actual type of object

toString and instanceOf

* Override the toString method to yield a String that describes the object’s state.
* The equals method checks whether two objects have the same contents.
* If you know an object belongs to a given class, use a cast to convert the type.
* The instanceOf operator tests whether an object belongs to a particular type.

Interfaces Summary

* Inteface type contains the return types, name, and parameter variables of.
* Interface provides no implementation.
* By using an interface type for a parameter variable, a method can accept objects from many classes.
* The ‘implements’ reserved word indicated which interfaces a class implements.
* Implement the comparable interface so that objects of the class can be compared.

GUI Components

* JComponent
* JFrame – to create the frame of the graphical interface, must set it visible, not visible by default, have to give it a size, by default it is very small (shows in the top left corner)
  + We add one button it would take up the whole frame, if we add another one it will go over the first button.
  + Border layout is the default layout for the frame
* JPanel – after the frame, set the panel in the frame, can set the certain layout format, by default it is flow layout (pushed in from right to left, if it gets crammed on the first line, it goes to the second. When we resize it bigger if there is space, they will move to the first line.
  + Menubar
  + Contentpane
    - Components
    - Container
      * components
* JTextField
* JScrollPane
* JButton
  + Put buttons in a button group
* Radio buttons
* Check Boxes
* Combo Boxes
* MenuBar
* JTextArea – side, .setText = will set the text to that area, append will at it to the end.
* JTextComponent
* JLabel
* Event Listener
  + Must know the EVENT SOURCE
  + ActionListener interface has one method – action listerner class is inside the class that uses it (inner class)
    - Must create the object ActionListener before using it.
    - Method is ActionPerformed
    - It has an inner class that performs the action clicked.
* Layout Types
  + Grid layout (4,3) 4 rows, 3 colomns, equal size
    - Uses x, y coordinates to specify
  + Borderlayout
    - North, South, East, West
  + BoxLayout
    - Single row or column, horizontally and vertically
  + CardLayout
    - Controlled by a combo box
  + FlowLayout
    - Default layout manager for every jpanel, single row
  + GridBagLayout
    - Grid of cells can span more than one cell, can have different heights and grid colomns can have different widths
* Graphics
  + paintComponent
  + shapes: Squares, rectangles, circles, ellipses, lines and text
    - drawRect, drawOval, and drawLine.
* JSliders
* Javax.swing.timer
* Mouse Events
  + mousePressed
  + mouseReleased
  + mouseClicked
  + mouseEntered
  + mouseExited

Threads

* Thread
  + A program unit that is executed independently of other parts of the program
  + There is no order in which threads are ran
* Running threads
  + The JVM runs each thread in the program for a short time
  + It gives the impression that it is all running at the same time, parallel execution
  + When implementing a thread it must implement the runnable interface in the class
    - Then create a run method that preforms the tasks
  + You must create an object of your Runnable interface in the class it is implemented in
    - Runnable r = new MyRunnable();
    - //Runnable interface
    - //MyRunnable 🡨 the class
  + Then create the object thread
    - Thread t = new Thread(r);
  + Then call the start method to start the thread
    - t.start();
* Terminating threads
  + Thread is terminated when the run method is terminated.
  + Instead of using the ‘stop’ method, instead we notify the thread using the ‘interrupt’ method
    - t.interupt();
      * It doesn’t cause the thread to stop, instead it sets a Boolean value in the threads data structure
  + The interrupted thread should release resources and clean-up, then exit
  + When we use the ‘sleep’ method, it throws an ‘InterruptedExeception’ when a sleeping thread is interrupted
  + The java language does not force a thread to terminate when it has been interrupted
    - It is completely up to the thread what it does when it has been interrupted
    - We us the interrupt method to get a threads attention
* Race Conditions
  + When threads share a common object, they can conflict with eachother and lock up the thread
  + Locks
    - Used to control threads that manipulate shared resources
    - We create a lock object, the lock interface and several classes that implement it
      * ReentrantLock 🡨 most commonly used lock class
    - A lock object is added to a class whose methods access shared resources
      * Private Lock balanceChangeLock; 🡨 declaration
      * (in the constructor we create the object of the lock)
        + balanceChangeLock = new ReentrantLock();
    - Code that manipulates shared resources is surrounded by calls to ‘lock’ and ‘unlock’
      * balanceChangeLock.lock();
      * balanceChangeLock.unlock();
      * If the code between the two calls throws an exception, the call to unlock never occurs.
        + To overcome this issue, we place the unlock into a try-finally clause
        + When a thread calls lock, it owns the lock until it calls unlock

A thread that calls lock while another thread owns the lock is temporarily deactivated

* + - * + Thread scheduler periodically reactivates thread so it can try to acquire the lock

Eventually, waiting thread can acquire the lock

* + Deadlocks
    - Occurs if no thread can proceed because each thread is waiting for another to do some work first
      * We use a condition object to overcome this
      * Condition objects allow a thread to temporarily release a lock and to regain the lock at a later time
        + Each condition object belongs to a specific lock object
        + Use the newCondition method of the lock interface
      * Calling AWAIT
        + Makes current thread wait
        + Allows another thread to acquire the lock object
  + Signal/SignalAll
    - To unblock, another thread must execute signalAll on the same condition object
    - signalAll unblocks all threads waiting on the condition
    - signal randomly picks just one thread waiting on the object and unblocks
      * Signal can be more efficient, but you need to know that every waiting thread can proceed
    - We recommend to call SignalAll
* Avoiding Deadlocks
  + A deadlock occurs if no thread can proceed because each thread is waiting for another to do some work first
  + Calling ‘await’ on a condition object makes the current thread wait and allows another thread to acquire the lock object
  + A waiting thread is blocked until another thread calls ‘signalAll’ or ‘signal’ on the condition object for which the thread is waiting

Network programming w/ java

* Internet Protocol
  + Internet
    - A worldwide collection of networks, routing equipment, and computers
    - Uses a common set of protocols to define how the parties will interact with each other
  + Data Transmission consists of sending/receiving streams of zeroes and ones along the network connection
  + Two types of information:
    - Application data – information one computer wants to send to another
    - Network protocol data – describes how to
      * Reach the intended computer
      * Checks for errors & data loss in the transmission
  + Network Protocol: rules for protocol data
    - Most Common: Internet Protocol (IP)
      * Developed to enable different local area networks to communicate with each other
      * Has become the basis for connecting computers around the world together over the Internet
  + Packets
    - IP breaks large chunks of data up into more manageable packets
    - Each packet is delivered separately
    - Each packet in a larger transmission may be sent by a different route
    - Packets are numbered
    - The recipient reassembles them in the right order
  + Transmission Control Protocol (TCP)
    - IP does not notify the sender if data is lost or garbled
    - TCP is higher level protocol
      * Used by Internet services – WWW and email
      * Bypassed by streaming media service for highest possible throughput
  + Port Numbers
    - IP uses port number
      * Integer between 0 and 65535
      * Sending program must know port number for receiving program
      * Port number is included in the transmitted data
  + CONTENTS OF TCP PACKET
    - IP of recipient
    - Port number of recipient
    - Internet Address of recipient
    - Port number of sender
* Application level protocols
  + TCP/IP mechanism establishes an internet connection between two ports on two computers
  + Each Internet App. Has its own app. Protocol
    - Describes how data for that application are transmitted
  + Hypertext Transfer Protocol (HTTP)
    - Application protocol used for WWW
    - Web address is called a Uniform Resource Locator (URL)
    - URL is what you type after HTTP://
    - HTTP uses default port 80
  + HTTP (2)
    - HTTP and HTML are not the same
    - HTML is a document format, and describes structure of a document
    - HTTP is a protocol that describes the command set for web server requests
    - Web browsers
      * Know how to display HTML documents
      * And how to issue HTTP commands
    - Web servers
      * Know nothing of HTML
      * Vaguely understand HTTP and how to fetch requested items
    - Post Office Protocol (POP)
    - Simple mail transfer protocol (SMTP)
* Client program
  + Tasks:
    - Establish a TCP connection to a server
    - Sends a request to the server
    - Prints the response
  + Sockets
    - A socket is an object that encapsulates a TCP/IP connection
    - There is a socket on both ends of a connection
    - Create a socket
      * Socket s = new Socket(hostname, portnumber);
    - It will throw a an UnknownHostException if it cant find the host
  + Input and output streams
    - Use the input and output streams attached to the socket to communicate with the other endpoint
      * InputStream instream = s.getInputStream();
      * OutputStream outstream = s.getOutputStream();
    - When you send data to outstream, the socket forwards them to the server
    - The socket catches the servers response and you can read it through instream
    - When you are done communicating with the server, close the socket
      * S.close();
  + Scanners and Writers
    - InputStream and OutputStream send and receive bytes
    - To send and receive text, use a scanner and a writer
      * Scanner in = new Scanner(instream);
      * PrintWriter out = new PrintWriter(outstream);
    - A printwriter buffers the characters and only sends when the buffer is full
      * Buffering increases performance
    - When sending a command, you want the whole command to be sent now
      * Flush the buffer manually:
        + Out.print(command);
        + Out.flush();
* Server program
  + Server socket
    - Server waits for clients to connect on a certain port
    - To listen for incoming connections, use a server socket
    - We provide the port number when constructing the server socket
      * ServerSocket server = new ServerSocket(port);
    - We use the accept method to wait for client connection and obtain a socket
      * Socket s = server.accept();
* URL connections
  + Java library class URLConnection provides convenient support for the HTTP
  + Construct a URL object from a URL starting with HTTP of FTP
  + Then use the URL’s openConnection method to get the URLConnection
  + Call the getInputStream method to obtain an input stream
  + Turn the stream into a scanner in the regular way

SQL

* Relational Databases
  + Stores information in tables
  + Each table column has a name and data type
  + Each row corresponds to a product
  + Column headers correspond to attributes of the product
  + All items in the same column are the same data type
* SQL
  + Structured Query Language
  + Command language for interacting with databases
  + Non case sensitive
  + Primary Key:
    - A column (or combination of columns) whose value uniquely specifies a table record
    - Not all databases need a primary key
    - You need a primary key if you want to establish a link from another table
  + Foreign Key:
    - A reference to a primary key in a linked table
    - Foreign keys need not be unique
* SQL Data types
  + Integer or INT
  + Real
  + Double
  + Decimal(m, n)
  + Boolean
  + Varchar(n)
  + Character(n) or char(n)