**Exceptions**

* Purpose of catching and handling thrown exceptions to create robust software
* Methods declare exception throwing with “throws”
* Exceptions are handled with a try and catch block
  + The block can have multiple catches going from specific to generic exceptions, such as ArrayIndexOutOfBoundsException to RuntimeException to Exception
* A method throws the exception when something occurs with the “throw” command followed by the exception, and then is poped from the method to the caller until the exception is handled
* The Exception hierarchy is Object -> Throwable -> Exception -> RuntimeException (the majority exist here)
* There are two types of Exceptions
  + Checked are compiler checked exceptions like a file buffer will throw an error saying that its exception is not handled
  + Unchecked are runtime exceptions like a NullPointer or ArrayIndex exception and are a choice by the programmer on how handle the exception
    - Are most likely logical or algorithmic issues and can not be predicted, therefore try and catch are used as a precaution

**Generics**

* The use of generics provides reliability and readability
* Benefits of making class and objects generic
  + Detect errors at compile time
  + Type checking that prevent errors from occurring at runtime
  + Removes casting
* Type inserted into the generic must be the type not the primitive such as Integer instead of int
* Eliminates duplicate code as all class methods can be made to work for generics rather then functions for each type
* Generic types can be extended to include extra functionality by using the extends use inside the generic declaration

**Collections**

* The collections framework provides interfaces and classes to structure both homogenous and heterogenous data depending on the situation and type of collection
* Can be iterated through with an iterator class
  + Collections have three branches
    - List which is ArrayList and LinkedList which are as follows
      * Ordered Collection
      * May Contain duplicated elements
      * Funtions for list (collections-1 pdf page 26)
      * ArrayList is best for adding and removing from the end of the list or random access with the index
      * LinkedList is best for adding and removing anywhere in the list
    - Queues
      * Offer(element) inserts into the queue
      * Poll() retrieves and removes the head of the queue
        + Returns null if empty
      * Remove() retrieves and removes head of queue
        + Throws exception if empty
      * Peek() retrieves but does not remove head
        + Returns null if empty
      * Element() retrieves but does not remove head
        + Throws exception if empty
      * Deque is similar to a linkedlist
    - Set and HasSet which are as follows
      * Similar to the mathematical set, no set order
      * Does not allow Duplicate items (make use of comparator class)
      * Insertion order is not preserved
      * HashSet no order
      * TreeSet implements SortedSet to allow ordering
        + Makes use of natural ordering

Numbers ascending/descending

Capital letters before lower case

Done by unicode

* + - * LinkedHashSet allows insertion order preservation
  + Also included are Maps and HashMaps
    - Maps are a key and value pair structure, where keys cannot be duplicated but values can
    - HashMap is good for locating values, inserting and deleting
    - TreeMap allows for sorting and and traversing keys in an order

**Internationalization**

* Allows for code extension across multiple markets
* Textual messages match local language
* New languages do not require recompilation
* Data can be targeted to specific demographics
* Application can be localized quickly
* Three Main classes Locale, NumberFormat and DateFormat
  + Locale provides and gets local country and language
    - Called the usual class new Locale()
  + DateFormat provides the local date and formats according the to the local formatting type
    - getDateInstance() -> default date formatting
      * constructors for specific formats or locales
    - getDateTimeInstance() -> default date time formatting
      * constructors for specific formats or locales
  + NumberFormat provides the local number/currency formatting to the local type
    - getInstance() -> number format
    - getCurrencyInstance() -> currency format
    - getPercentInstance() -> displaying local percentages
  + TimeZone class provides the ability to get the local timezone or a specific timezone