fiveable 💻 AP COMPUTER SCIENCE A CRAM CHART // @thinkfiveable // http://fiveable.me

Primitive Types
Unit 1 ↓

- System.out.println(String textToPrint): ends with line break
- System.out.print(String textToPrint): no line break
- System.out.printf: allows for easy formatting and concatenation of strings
- String - in “”, text, not primitive data type
- boolean - true/false (no capitalization), represents truth value
- int - represents an integer value with 4 bytes of storage between Integer.MIN_VALUE and Integer.MAX_VALUE
- double - represents a decimal up to about 15 places of accuracy (but take care for rounding/overflow when practically coding)
- final - cannot change
- +, -, and * work how you expect them to
- / returns a decimal only if one of the values is a double, else it gives the whole number answer, while % gives the remainder
- Order of operations: (), %*/, +- from left to right, just like PEMDAS
- = is assignment, == is equality
- +=, -=, *=, /=, %= perform the operation on a number type and reassigns the variable value (compound assignment)
- ++ and —— allow you to add or subtract a variable value by 1 and stores the value after
- (int) (double x) and (double) (int x) can convert between number types
- (int) will lose precision while (double) will add precision
- Rounding (int)(x+0.5) if pos, (int)(x-0.5) if neg
- Integer and Double wrapper classes part of java.lang
- Integer(int value): makes a new Integer object from an int value
- int intValue(): returns the value of an Integer as an int
- Double has corresponding methods to Integer
- Autoboxing: primitive -> wrapper
- Unboxing: wrapper -> primitive
- Math.lang contains all static methods
- int abs(int x): absolute value (also double abs(double x))
- double pow(double base, double exponent)
- double random() RNG between 0, inclusive, to 1, exclusive, can be manipulated for rand in range

Using Objects
Unit 2 ↓

- An object is an instance of a class, which provides the attributes and methods (functions) that the object will have
- methodName(dataTypeOne parameterOne, dataTypeTwo parameterTwo, ...) is the method signature, which can be overloaded by having different versions of the parameter list
- This is call by value, initializes parameters with copy of actual
- me(parameterListIfNecessary)) make a new object, if parameter list empty, all instance variables set to null
- Method declaration: scope (static if necessary) returnType methodName(parameterList)
- Scope is public or private (depends on if you want the method to be accessible by other classes)
- Static means that the method is not keyed to any instantiation (object) of the class, while non-static methods are
- The returnType can be void, which means that the method does not return anything, if the method returns something, you must state its type in the declaration
- Calling a static method of the same class: methodName(parameters)
- Calling a non static method of the same class: objectName.methodName(parameters)
- If the method comes from another class, put ClassName. before the method call
- String methods do not change the String object
- Can concatenate string objects and primitive objects using += or +, if concatenating non-primitive object, implicitly calls toString()
- String indices from 0 to length-1
- String(String str): makes a new string with same characters
- int length(): returns length of string
- String substring(int from, int to (optional)): returns substring from index from to one before index to, if to not present, then to end of string
- int indexOf(String str): returns index of first occurrence of str, -1 if not found
- boolean equals(String other): returns true if strings =
- int compareTo(String other): returns the “difference” between 2 strings
- substring(index,index+1) returns the character at index

Boolean Expressions and if Statements + Iteration
Unit 3 + 4 ↓

- ==: equality, !=: inequality
- </>: less/greater than, <=/>=, less/greater than or equal to
-
  ```
  if (condition) {
      doThis;
  } else if (condition) {
      doThis;
  } else {
      doThis;
  }
  ```
- &&: and, ||: or
- == only refers to the same object, equals method for equivalent objects
- while (conditionThatIsNotMet) {
- thingThatWillRunUntilCondMet;
- }
- for (initialization; condition,; increment) {
- thingThatWillRunUntilCondMet;
- }
- Usually header (int i=0;i<timesToRun;i++)

Writing Classes
Unit 5 ↓

-
  ```
  scope ClassName {
      private instVarType instVar
      public
  ClassName(parameterList) {
          constructor; //can be
  overridden, if no parameter list,
  set to default
          this.instVar = instvar;
      }
      public instVarType getVar() {
          return instVar;
      }
      public void setVar(instVarType
  newInstVar) {
          instVar = newInstVar;
      }
  }
  ```

Arrays
Unit 6 + 7 ↓

- dataType[] arrayName = new dataType[numItems]
- fixed number of items
- initialized with 0, 0.0, false, or null
- index from 0 to arrayName.length - 1
- enhanced f
- or loop uses header (dataType name: arrayName) where name will be each item in the array

ArrayList: Unit 7 ↓

- import java.util.ArrayList
- ArrayList<E> name = new ArrayList<E>();
- variable number of items
- only contains objects
- int size(): size
- boolean add(int index (optional), E obj), adds obj to end if no index, else at index value and shifts items at index and higher to the right, returns true
- E get(int index): gets element at position index
- E set(int index, E obj): replaces and returns former object at index with obj
- E remove(int index): removes and returns item at index, shifts rest of items to left
- traverse like arrays
- don’t add or removed while traversing

2D Arrays
Unit 8 ↓

- 2D arrays are stored as arrays of arrays.
- Arr[rows][columns] is how to access them
- To traverse, you use nested loops
- Everything you do with a normal array can be applied to 2D arrays
- When searching, each row must be accessed then searched.

Inheritance: Unit 9↓

- A class hierarchy can be created by putting classes under a single superclass and classes below that being subclasses
- subclasses, can draw upon the existing attributes and behaviors of the superclass
- Using “extends” you can create this hierarchy
- Constructers have to be created under each subclass(using “super” you can call the superclass)
- Any method that is called must be defined within its own class or its superclass.
- Overidering occurs when a subclass method signature is the same as the superclass
- Using super can be used to call methods in the superclass
- Changes in the superclass reflect onto the subclasses

Exam Tips
FRQ/MCQ ↓

- Comment the code you see! This way if you ever have to go back you don’t have to fully reread the code each time!
- Don’t Rush! You might miss something that way!
- If there's a large chunk of code you don’t understand, break it down!
- Don’t be afraid to take some time beforehand to write some pseudocode so that you can code faster!
- Write Method Header: Remember scope, (static if needed), return type, and proper parameters with input data types
- Identify: Just say what is needed to be added
- Describe: Add depth to your identifications, with features, characteristics, type and scope, input types, and return types

Recursion: Unit 10↓

- Recursive methods contain at least one base case, which halts the recursion, and at least one recursive call.
- Each call has its own local variables
- Binary search can be more efficient and starts in the middle of a sorted array
- MergeSort is a recursive method and can be used on ArrayList