CONTENT REVIEW GUIDE

Johns Creek High School
Cmdr. Raymond T. Schenk
v1.0 11 December 2018

This work is the student powered! My amazing students have come together to help assemble a work of information that is aligned by our course standards and put together to provide a fundamental platform for equipping future students at mastering the curriculum for this course.

Their hard work and leadership has made this entire guide possible, and they deserve the credit for amassing a wall of information, in service to their peers. I am forever grateful to their ability to leave a legacy for those come behind them.

AP COMPUTER SCIENCE A

V1 COMPUTING IN CONTEXT

These general topics will be reviewed and discussed across our projects and discussions:

CTAE Review: Professionalism

Definition of *professionalism*

1: the conduct, aims, or qualities that characterize or mark a profession or a professional person

https://www.merriam-webster.com/dictionary/professionalism

Things that show a lack of professionalism:

- Casual clothes, lack of effort in appearance
- Inappropriate language or volume of voice
- Missing deadlines with work
- Lack of motivation towards job responsibilities
- Blaming others for your mistakes
- Missing work
- Disrespectful behavior
- Unethical behavior or tolerance thereof

PROFESSIONALISM SHOULD BE MAINTAINED AT ALL TIMES IN THE WORKPLACE.

Things that indicate strong professionalism:

- Strong work ethic
- Creativity
- Working towards the mission of the company or organization
- Cooperation
- Service to others
- Self-regulation
- Motivated productivity and ambition
- Strong ethical performance
- Humility
- Proper Planning
- Respect for others

SYSTEM RELIABILITY

Reliability of any software system is a complex topic. Modularity is a key component of scalability and reliability. Separation of Concerns (SoC) is a topic we visit often, which involves decoupling program modules into separate "concerns." Specifically, data models should be separated from the business logic that controls them, which should be separated from the views and view models that display results and queries to end users.

Well-designed systems will also provide for data redundancies and backups, as well as intrusion and malicious intent mitigations. System security needs to be holistically applied in a manner that minimizes interference with operations.

PRIVACY

Privacy is a perpetual issue in today's connected environment. All major social media sites are now facing government regulation (interference) for their inability or otherwise unwillingness to protect the privacy of their end-users. These types of ethical dilemmas will remain at the forefront of our culture and economy for years to come as we adjust to a new digital age.

LEGAL ISSUES AND INTELLECTUAL PROPERTY

With China stealing and replicating volumes of IP, intellectual property discussions and laws will continue to dominate many computer science discussions for years to come as well. China is certainly not the only nation involved in technical espionage. Ethics also arise from the marketing of personal information and online patterns by social media giants like Google, and Facebook as noted above.

Social Ramifications of using computers still continue to emerge in new ways daily. Some are amazingly productive, and some are quite damaging. Cyber-bullying sits in sharp contrast to crowd-sourcing genomic research to fight certain formations of cancer cells. As a society we must deal with an ever changing and ever-accelerating landscape of technical innovations.

I OBJECT-ORIENTED PROGRAM DESIGN

PSEUDO CODING

Advantages of pseudocode:

- Pseudocode is understood by the programmers of all types.
- It enables the programmer to concentrate only on the algorithm part of the code development.
- It cannot be compiled into an executable program.

Pseudocode is basically just writing down the logic of your solution to a specific coding using plain English. Pseudocode is a useful thing to learn because it focuses on the building block concepts of programming languages without you having to worry about your word choice

FLOW CHARTING

Symbol	Purpose	Description
	Flow line	Used to indicate the flow of logic by connecting symbols.
	Terminal(Stop/Start)	Used to represent start and end of flowchart.
	Input/Output	Used for input and output operation.
	Processing	Used for arithmetic operations and data-manipulations.
	Decision	Used to represent the operation in which there are two alternatives, true and false.
	On-page Connector	Used to join different flowline
	Off-page Connector	Used to connect flowchart portion on different page.

	Predefined Process/Function	Used to represent a group of statements performing one processing task.

ABSTRACTIONS IN USE

Any method is an abstraction. A method is a named series of instructions that can be called repeatedly. It can have a return value (function) or simply return nothing - void (procedure). Either way, the named method can also accept input parameters. In the definition of the method, these parameters are called ***formal parameters***, because they are merely variable names scoped to the internal domain of the method. The ***actual parameters*** are the values passed into the method via the formal parameters whenever the function is called.

If a method calls another method, this is tiering methods, or a "higher order" of abstraction. When we import libraries of classes and other methods from other programmers to use, we are in essence standing on the shoulders of those who have gone before us. These layers of abstractions are what make programming as powerful as it is today.

CLASSES

Classes are custom data types, a "recipe" for an object.

They become objects when we instantiate them. This occurs when we invoke the custom data type in our main() and assign a variable name, and then call the constructor of our choice.

Classes have fields (HAS-A)

And Methods (Functions (return a value) or Procedures (returns void))

Child classes embody IS-A relationship to their parent.

i.e. **Boy** IS-A **Human**. **Girl** IS-A **Human**. **Prius** IS-A **Car** (Barely)

Class names use Pascal Notation

Internal reference term is reserved word: ***this***

FIELDS

Private, Protected or Public [unmodified is package public]

Camel Case names

Fields are the data storage of the class. Can hold primitives or other objects.

METHODS

Work Horses of a class. Give a class the ability to *do things*.

Procedures or Functions

Named in Camel Case

Each method should do one thing well.

System out printing in a method not specifically designed for output is an example of a side-effect (Bad except for troubleshooting)

Special Cases:

Constructors – Always public (by default) – must initialize ALL FIELDS and perform any other initialization functionality.

Getters & Setters – Also public. Provide rights and permissions checking, then return field value as requested (getters), or replace field value (mutators/setters).

ENCAPSULATION

What goes on in a class, stays in a class.

Encapsulation protects the inner-workings of our class, to abstract away the details. When we share our methods and classes with others, we do not expect them to have to worry about how we achieve our method functionalities.

We control what they send us, and what we send them back. We do the work internally.

We inspect their permissions in getters and setters, so that we know that only authorized personnel are mutating our instance variables. We provide public property wrappers for our fields for just this purpose, or at a minimum, getters and setters.

We let our children in only when we need to as protected access control recipients.

We default to private level encapsulation for all but our constructors and public getters and setters.

When we wish a field to be READ ONLY, we do not supply a setter, and in the few instances we want to allow no read, but we do need to allow a setter, we make out fields WRITE ONLY by providing only a setter.

CONSTRUCTORS

Our classes must have proper initializations. This is why we call their parent constructors in ours whenever appropriate. (Normally).

We provide to our parent classes any field values they need and supply the rest to our internal fields. When none are provided, as in default and partial constructors, we handle all internal initializations for all fields ourselves in the appropriate constructors.

We never leave it to the compiler to do our work for us, because we are not sloppy programmers.

We provide users of our libraries (often we are our own users), with a selection of constructors to make their lives easier. This is because we have *empathy* for their work. We use the best design-thinking strategies to develop reasonable rules-of-the-road when it comes to instantiating our classes into objects and implementing our interfaces (contracts).

All calls to parent are calls to "Super()" – enclose parameters to super as needed for initialization.

Constructors can OVERRIDE any values to parent if needed. Pass updated values in call to super.

II PROGRAM IMPLEMENTATION

DATA TYPES

Byte – 8-bit signed two's complement integer. -128 to 127

Short – 16-bit two's complement Integer. -32,768 to 32, 767

Int – 32-bit two's complement Integer. -2^{31} to 2^{31} -1

Long – 64-bit two's complement Integer. -2^{63} to 2^{63} -1

Float – Single precision 32-bit IEEE 754 floating point.

Double – Double precision 64-bit IEEE 754 floating point.

Char – Single 16-bit Unicode character. Min value is '\u0000' to '\uffff' (65,535 inclusive)

Boolean – true or false. Represents a single bit of information, but "its 'size' isn't something that's precisely defined." (Oracle at its best.)

****new keyword is never used in defining primitive data types because they are built into the language, not objects created from a class.

THERE ARE FOUR KINDS OF REFERENCE TYPES IN JAVA:

Class types (includes enums) – Classes as we learn them

Interface types – Interfaces and "Annotation types" – Interfaces as we learn them

> Programs can use interfaces to make it unnecessary for related classes to share a common `abstract` superclass or to add methods to `Object`.

Type variables – an unqualified identifier used as a type in a class, interface, method and constructor bodies. Used in generics.

Array Types – Arrays declared to hold specific data types. Number of [] determines how deep the array nesting goes. i.e. int[] integerArray is an array of integers.

DECLARING CONSTANTS

Static means instead of having space allocated for a field in each object, only one is ever created. **Final** means a value can never be changed ***after*** initialization.

Hence, **static final** means only one instance is created that can never be changed after it is initialized, hence it is a constant.

DECLARATIONS

A declaration allocates memory, sets a token to point at the memory (Which if it is a primitive or reference variable is stored on the stack), puts this all in the table of tokens and what they point to, then assigns the value determined on the right-hand side of the assignment operator.

JAVA MEMORY (GROSSLY SIMPLIFIED)

The Java stack is for thread specific data. Each thread gets its own stack inside the JVM.

When a thread enters a new block called a stack frame is created inside that threads stack memory.

Stack memory is less than heap memory.

Objects and arrays go on the Java heap. Heap memory is common to all threads.

Garbage collector is triggered when max size of heap is hit. Programmers have ZERO control of garbage collector. “Managed code.”

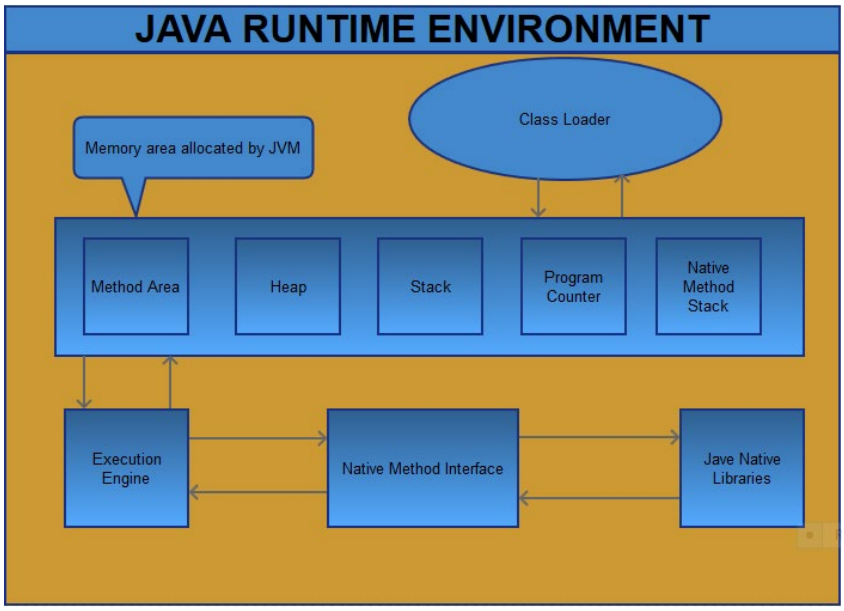
Heap has two logical portions: Young Generation and Old Generation

New Generation – newly created objects

Old Generation – older objects after a set amount of time (surviving several rounds of garbage collection) get “promoted” to old generation.

https://javabeat.net/jvm-memory/

It's more complicated, but that is for another class:



CONTROL STATEMENTS

Sequence – Put your instructions in the correct algorithmic order.

Selection – If/Else, Switch

Iteration – Looping: for loops, enhanced for-loops, Do-while, While-Do

III PROGRAM ANALYSIS

TESTING AND DEBUGGING

Setting break points – suspend flow execution to analyze program state

Daemon code commands to see results, flow location entry/exit (SOP)

Watch Lists – viewing variable values in debug mode

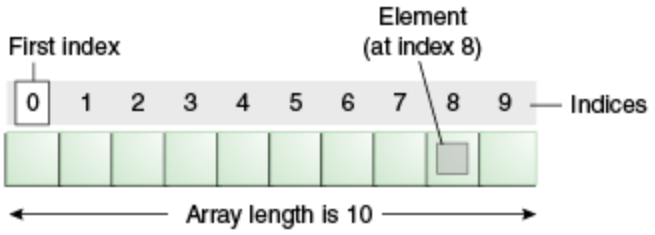
TRACING CODE

Tracing code is the act of manually following code and determining values as you go.

IV STANDARD DATA STRUCTURES

ARRAYS

Arrays



Each item in an array is called an element.

Arrays are containers holding fixed numbers of values of a **single** data type*.

Rows can vary in length because in Java a 2-dim array is an "Array of arrays."

Arrays can hold objects. Create an array of the object type. If a parent object, then the array can hold polymorphic child objects in the same array. (Cool trick)

LISTS

List is a generic interface in Java.

It is AN INTERFACE. A Generic INTERFACE.

Hence, I must be IMPLEMENTED.

This is why on the AP Exam you will see something like this:

List<myClass> myList = new ArrayList<myClass>();

The reference variable myList is of type List<myClass> which is an interface.

Therefore, the right-hand operator must be a class that implements the List interface, which the class ArrayList does.

V STANDARD ALGORITHMS

TRAVERSALS

Traversals are moving through an ADS (Abstract Data Structure) like a tree.

INSERTIONS AND DELETIONS

CRUD – Create, Read, Update, and Delete are operations you will need to perform on all ADS.

Adding record appropriately is critical, as is removing the correct entry in the ADS.

SEARCHING: SEQUENTIAL AND BINARY

Sequential searching is self-explanatory. Start at the beginning and touch each element in any collection, looking for the value you need to match.

Binary searching only works on an ordered set of whatever you are searching. Think “phone book” looking for a name. Split solution in half, determine which half must contain the correct value, then search again. This is literally exponentially faster. Doubling the search set only adds a single division by two to a binary search.

SORTING

Big O Notation – Big O() is the **worst**-case analysis of an algorithm, typically used in computer science to compare sorting algorithms or major traversals of ADS (Abstract Data Structures).

- Bubble, Insertion, Selection and Quick sorts are all O(n^2)
- Merge is O(nlog(n)) (Faster for larger data sets because of recursion)

Omega Notation – Omega notation is **best**-case performance, in the same light as O().

- Bubble, Insertion Ω (n)
- Selection Ω (n^2)
- Merge, Quick, Ω (nlog(n))

You only need to know Insertion, Selection, Merge:

Sort	O( )	Ω( )
Insertion	n^2	n
Selection	n^2	n^2
Merge	n(log(n))	n(log(n))

EXTENDED LEARNING

Software Development Life-Cycle (SDLC)

Waterfall – oldest, slowest, very inflexible, Linear

Others: Prototyping, Spiral

Best: Agile – Cyclical and Iterative

Individuals and Interactions over processes and tools

Working Software over comprehensive documentation

Customer Collaboration over contract negotiation

Responding to Change over following a plan

Runs in Sprints, is Iterative, Incremental and evolutionary

Scrum is a specific software development agile methodology. 3-9 members. 2-4 weeks max sprints. Daily scrums are 15-min re-planning meetings.

THE FIRST ABSTRACTION: HOW COMPUTERS REPRESENT DATA

ANSI - American National Standards Institute – American non-profit organization that develops and maintains standards for industry.

ASCII – American Standard for Information Interchange. Written and developed/maintained by ANSI. ANSII was the world's answer to defeat IBM's EBCDIC

ASCII DIGITS

011 0001	1
011 0010	2
011 0011	3
011 0100	4
011 0101	5

ASCII CHARACTERS (UPPER CASE)

100 0001	A
100 0010	B
100 0011	C
100 0100	D
100 0101	E

ASCII CHARACTERS (LOWER CASE)

110 0001	a
110 0010	b
110 0011	c
110 0100	d
110 0101	e

ASCII CHARACTERS (SPECIAL CASE)

000 0000	Null
111 1111	Delete

- Computers use 7 bits for ASCII (8 bit is 0) – Fits in a single byte
- Extended ASCII uses the extra bit to add more characters but contains ASCII perfectly.
- UNICODE uses between 1 and 4 bytes, *usually* containing a character. It contains all of ASCII and a lot more characters and combinations of accents. It ***does not*** contain every character of every language, but many say it tries to do so.

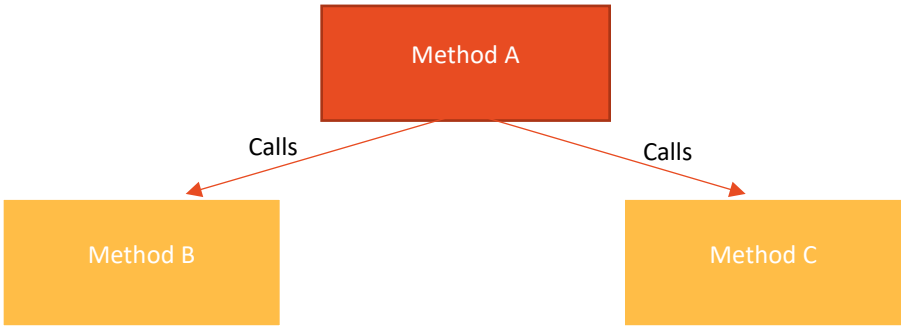
Storage Area Network (SAN) – an array of hard drives that ship only with low level block I/O capabilities. Must be formatted and initialized under a specific operating system's disk management protocols. These units are typically very large and expensive.

Network Accessible Storage (NAS) – more affordable hard drive space that can be accessed by users on a network. Many home and small offices use RAID-based NAS devices for storing and/or backing up valuable files.

Virtual Private Network – A network that allows users to access its resources via an encrypted session. VPNs provide security to an organizations network but can be considered a vulnerability to networks hosting users who use them.

P4: ANALYZING PROBLEMS AND ARTIFACTS

For CREATE, the college board will require a small pyramid tier of abstractions, which can be visualized in the following diagram:



IN THIS DIAGRAM, A SINGLE METHOD CALLS TWO OTHER METHODS. OF THE TWO OTHER METHODS, THE EACH SHOULD CONTAIN CALCULATIONS AND FLOW-CONTROL LOGIC. ADDITIONALLY, ONE OF THEM (B OR C) MUST BE USED REPEATEDLY IN THE PROGRAM TO "HELP REDUCE COMPLEXITY OF THE PROGRAM."

ENCRYPTION/DECRYPTION

Encryption requires ***plaintext***, a ***cipher***, and a ***key.***

Decryption requires ***ciphertext***, the ***cipher used for encryption***, and a ***key***.

Keys can be ***symmetric***, or ***asymmetric*** (public private)

GENERAL TERMS

Call - location in the code were a method is called.

Invoke - to activate or begin a function in a line of code

Variables - memory location with a name token used to store a specific data type.

Reference Variables - data type that points to an object in memory

Data Type - the classification of what data is being stored. Includes primitives or custom (objects).

Return Value - the value a method returns.

Programming Language – a formal language made up of a grammar, syntax, and a set of instructions used to produce various kinds of output.

Software Reuse - the process of creating software systems from existing software. (Leveraging previous abstractions)

Pascal Notation - naming in which the first letter in each word in a compound name is capitalized. Class and Interface names use Pascal notation.

Camal Case - naming in which the first letter in each word in a compound is capitalized ***except for the first letter in the name.*** Variable and method names use Camel Case, aka Camel-hump naming.

Big O Notation – Big O() is the **worst**-case analysis of an algorithm, typically used in computer science to compare sorting algorithms or major traversals of ADS (Abstract Data Structures).

- Bubble, Insertion, Selection and Quick sorts are all O(n^2)
- Merge is O(nlog(n)) (Faster for larger data sets because of recursion)

Omega Notation – Omega notation is **best**-case performance, in the same light as O().

- Bubble, Insertion Ω (n)
- Selection Ω (n^2)
- Merge, Quick, Ω (nlog(n))