**Dylan Gallup, CS260, MIDTERM CHEATSHEET**

**READINGS**

**-1. Intro to oop: every object has a state, behaviors (methods). Data encap: hide the details of how the data is stored. We want objects to be as independent of each other as possible. Easy to use in more than one application.**

**-1. Data encap: Combining data and how it's manipulated in one place.** This is achieved through the state (the private fields) and the behaviors (the public methods) of an object.**Only allowing the state of an object to be accessed and modified through behaviors.** The values contained within an object's state can then be strictly controlled.**Hiding the details of how the object works.** The only part of the object that is accessible to the outside world is its behaviors. What happens inside those behaviors and how the state is stored is hidden from view. Reasons: Keeping the state of an object legal. We can change the implementation of an object. Re-use of objects. The independence of each object.

**-2. Method sig: signature is the combination of the method name and the parameter list. Mostly inpart to overloading: the ability to write methods that have the same name but accept different parameters.**

**-2. Accessors and Mutators: enforce data encapsulation thought the use of accessors and mutators. Get and set. Accessor return the value of a private field. Return the same data type as their corresponding private field. Mutators to set a value of a private field. No return type, and accept a parameter that is the same data type as the corresponding private. Why: we want to hide the data of the object as much as possible. The extra buffer provided by these methods allows us to: change how the data is handled behind the scenes / impose validation on the values that fields are being set to.**

**-3. Static Method: by declaring a method using the static keyword, you can call it without first creating an object because it becomes a class method. Static methods are used for methods that do not need to access to an object’s state or only use static field.**

**-3. Events: an object that is created when something changes within a graphical user interface. Two key elements: the event source and the event listener. Note; that multiple listeners and event sources can interact with one another. For a similar set of components that perform the same type of action, one event listener can handle all the events. Similarly, a single event can be bound to multiple listeners, if that suits the program’s design (although that is less common).**

**-3. USER INT TIPS: Important: more intuitive the user interface the easier it is to use, and the easier is to use and the less expensive to use it. Tips and techniques:** Consistency, consistency, consistency; Set standards and stick to them; Be prepared to hold the line; Explain the rules; Navigation between major user interface items is important; Navigation within a screen is important; Word your messages and labels effectively; Understand the UI widgets; Look at other applications with a grain of salt; Use color appropriately; Follow the contrast rule; Align fields effectively; Expect your users to make mistakes; Justify data appropriately; Your design should be intuitable; Don’t create busy user interfaces; Group things effectively; Take an evolutionary approach;

**-4. UML associations: Association** is a [**relationship**](http://www.uml-diagrams.org/uml-core.html#relationship) between [**classifiers**](http://www.uml-diagrams.org/classifier.html) which is used to show that instances of classifiers could be either [**linked**](http://www.uml-diagrams.org/association.html#link) to each other or [**combined**](http://www.uml-diagrams.org/association.html#aggregation) logically or physically into some aggregation. Semantic relationship, or structural relationship.

-**4. ASSOCIATIONS; THO: “HAS A RELATION” due to the typical implementation in Java is through the use of an instance field. Bi-directional with each class holding reference to the other. Aggregation and composition are types of association.**

**-5. UML GENERAL: a binary taxonomic directed relationship between a more general classifier (superclass) and a more specific classifier (subclass). (Informally called a “IS A” relationship. ) Hollow triangle .**

**-5. Tho: java inheritance: an object is able to pass on its state and behaviors to its children. Superclass is the class that is being inherited from. There can be as many subclasses as you want same with levels. A subclass can only extend one superclass. Reuse code that has already been written.**

**GENERAL NOTES;**

**-operation is where the behavior to the class is defined. Implemented through methods.**

**-3 tired architecture: Presentaiton tier(user interface: input output) logic tier (handles the logic and evaluations, processes data between the two other layers. Data tier( information is sotred and retrieved from a database.)**

* **-id-ing classes: The verb phrases are shown in *italics* to show the potential operations**
* **The subject of each of the nouns is bolded to identify the potential attributes of each class/object.**
* Functional independence
* Each component does it own thing
* Circle class own draws circles
* Students deal with student info
* Its not apart of teh interfaces unless it is public
* Grammatical parsing Concept of written requirements then parse it out into its objects.

**Event source---GUI component with which the user interacts**

**Event listener -- object that is notified by the event source when an event occurs --the event listener receives an event object when it is notified of the event -- uses the object to respond to the event.**

**Event object**

**information about the event that occurred..includes a reference to the event source and any event-specific information.**

**Event Handler --the method that is delegated to handle the processing when the event is triggered**

**inheritance better understood.**

**constructors are not inherited. All derived class constructors in the chain will execute in order from top to bottom. When constructors are present in the class inheritance hierarchy, the ancestor constructors are executed first, with the highest ancestor being the first to execute. Derived class inherits the public and protected members of the base class. HOWEVER, the derived class cannot access any private members of the base class.**

**The Base class knows nothing about any derived classes. - Base class cannot then access any members of a child class**

**The Derived class however, is aware of its Base class. Derived class can access all public and protected members of the base class**

Method Overload:

Method signature: name of the methods/with the parameter list

Ex. name(int1,int2) --- Overload: name(double1)

NOT PART: the accessor and the return type

Normally, you overload every class.

Aka the constructor

Method Overriding:

Two methods with the same signature, and parent -> child relations

The parent will define a method then the child will override them

Abstract class: and override them

Interfaces. We override them in the implemented class.

Multiplicities --- Max and min and many; what are they

Composition --“Has a” relationship --You always have to instantiate the inner object when you make the containing class

Inheritance

“Is a” relationship

Code reuse and extension

Public and protected are only inherited by the child class

Derived class cannot access any private members of the base class.

Derived class: --- New version of the base class ---Dynamic binding…

Inheritance hierarchy

From top to bottom

When inheritance class are made

Every object that we create is inherited from the system object

extends

Constructors are not inherited.

Package private???

Default access modifier

Only java specific, not on test

Virtual methods(dynamic binding)

Abstract class: --You can’t instantiate an abstract -- You are required to override methods in the inherited classes

Interfaces:

Deals only with methods

Vs. abstract: an inherited class can only inherit from one class. While a class can implements as many interfaces as possible.

Defines the public interface

(when to use one othe the other)

Interfaces can be inherited by child class.

Some reading questions

Working with files;

Delimited file: records where the fields are separated by the delimiter (we have to break up the fields)

Serializable: