****

**Cubing a Pencil**

|  |
| --- |
| 1. ***Describe it****: A pencil is a long, cylindrical (sometimes hexagonal), device used to write on paper.* |
| 1. ***Compare it:*** *It is different than a pen because you can erase what you have written, which you cannot do with most pens* |
| 1. ***Associate it:*** *It makes me think of drawing because I like to draw with one. (It might make you think of just about anything, such as standardized tests or sharpening before class or your book bag. There is much room for personal thought at this level of thinking.)* |
| 1. ***Analyze it:*** *Pencils are generally made of wood surrounding and supporting a thin piece of graphite, which is the substance that marks the paper. It usually has a metal cylindrical clasp at the top, which holds an eraser.* |
| 1. ***Apply it:*** *It is used by rubbing the graphite tip onto a piece of paper. . .When an error occurs, the pencil is inverted in the hand, and . . .* |
| 1. ***Argue for or against it:*** *An argument in favor of pencils is that they are efficient tools for the taking of notes or writing a rough draft of something. It is easy to make corrections if you make a mistake, simply by inverting the pencil and using the attached eraser to rub out the error. They are also light weight and convenient to carry. Arguments against pencils are that they frequently need sharpening, the graphite tips break too easily, the graphite gets all over my clothes, I always leave it at home, etc. . .* |

**Name:\_\_\_Logan Stucki \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Class Period: 6\_\_\_ Date: 10/04/16**

**Cubing Concepts OOP Methods (Lesson 6a). Please evaluate each term according to the charts above & below**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Term/Concept** | **Describe It** | **Compare It** | **Associate It** | **Analyze It** | **Apply It** | **Argue For/Against It** |
| accessor/getter | **It is a method with a visibility modifier, return type, and a name.** | **It is like a modifier method because it accesses instance data. It is not like a modifier because it does not change instance data.** | **It makes me think of the claw games at stores. It goes and grabs something and then drops the thing down the chute to you.** | **Accessor methods are generally defined with a public modifier, the return type of the data we are accessing, a name that starts with the word “get”, and no parameters.** | **If we had an instance data GPA, an appropriate accessor method would be:**  **public double getGPA()**  **{**  **return GPA;**  **}** | **Arguments for accessor methods are that they help enforce encapsulation by requiring that client classes use them to access instance data. An argument against them is that they require an extra step to access instance data.** |
| assign | **It is how you give a value to a data type** | **It is like defining a variable, but you are filling the reserved memory space with data** | **It makes me think of filling a box with stuff** | **The variable to be assigned to is on the left while the data to be assigned is on the right** | **If we had to assign a variable it would look like this:**  **Bleck = “lol”;** | **Positive, otherwise variables would be useless** |
| complete constructor | **A constructor method that assigns non default values to all instance variables** | **It is like a constructor except for that it assigns values to all instance variables** | **It makes me think of making a custom PC** | **It takes a bunch of parameters** | **If we had a complete constructor, it would look like this:**  **public Foo(int bar, int foobar, int fooo) {**  **this.bar = bar;**  **this.foobar = foobar;**  **this.fooo = fooo;**  **}** | **Positive, because otherwise objects wouldn’t exist** |
| constructor | **It is how you create an object and assign values to the data types it contains** | **It is like a complete constructor, but does not necessarily have to assign custom values to all the instance variables** | **It makes me think of buying a PC in a certain custom color** | **It takes some parameters** | **If we had a constructor, it would look like this:**  **public Foo(int bar) {**  **this(bar, 0, 0);**  **}** | **Positive for the reason above** |
| default constructor | **A constructor that creates an object and assigns default values to data types** | **It is like a constructor, but it only assigns default values to instance variables** | **It makes me think of buying a generic PC** | **It takes no parameters** | **if we had a default constructor, it would look like this:**  **public Foo() {**  **this(0, 0, 0);**  **}** | **Positive for the reason above** |
| define | **It is how you reserve data space for a variable/data type** | **It is like assigning a variable, except you are making room in memory for the variable** | **It makes me think of pushing stuff around to make room for something** | **You have the variable data type on the left and the variable name on the right** | **If we had to define a variable, it would look like this:**  **int foo;** | **Positive otherwise variables would be useless** |
| instance variable | **A variable that is assigned a value upon object instantiation** | **It is like defining a variable, but instance variables are only used in objects** | **It makes me think of having different data in each drawer of a filing cabinet** | **At the top of the class under the class header but only initially assigned a data to in the constructor** | **If we had instance variables in the Foo object, it would look like this:**  **public class Foo {**  **int bar, foobar, foo;** | **Neutral, instance variables are mostly for program design and don’t add much functionality.** |
| instantiation | **How you create a new instance of an object** | **It is like defining a variable, but with an object** | **It makes me think of pushing tons of stuff around in order to make room for a massive crate of stuff** | **An object will be instantiated by calling a new object of that type** | **If we wanted to instantiate an object it would look like this:**  **new Foo(0);** | **Positive, otherwise objects would be useless** |
| mutator/modifier/setter | **A method that assigns a value to an instance variable** | **It is like a constructor, but it only assigns a value to (a) certain variable(s)** | **It makes me think of naming a baby** | **Used by calling the modifier method usually with one parameter** | **If we had a setter method, it would look like this:**  **public void setBar(int bar) {**  **bar = this.bar;**  **}** | **Neutral, these are mostly for program design and staying true to encapsulization** |
| object | **A class that can have more than one instance** | **It is like any normal class, except for that an object can have different values than another object of the same type** | **It makes me think of a giant crate full of smaller boxes** | **Used by defining and instantiating a new object** | **If we had an object, it would look like a normal class.**  **Public class Foo {**  **// constructors and stuff**  **}** | **Positive because you may want to have multiple classes with different data** |
| reference | **A memory of a variable/data type** | **It is like a variable, but it stores data to locate the data for the variable it’s referencing** | **It makes me think of a file that redirects somewhere else** | **Java looks at the reference to see where to locate data for that variable** | **A reference is what you see when you use a variable name in java.**  **Int foo** | **Positive because without it java would not know where the data is** |