## CS 5004 Module 7 Worksheet More Complex Program Design

Each worksheet will help you review some of the very important concepts that we have been learning in this class. It can be hard to learn the vocabulary that is used to discuss programming but it is very important for us to be able to communicate with each other, with future employers. In this worth, you should complete the following crossword puzzle which reviews the vocabulary we have learned thus far.

## 

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Across** | | | **4.** | if A = B and B = C, then A = C  **Transitivity** | | **5.** | an abstract data type in Java that has only one single abstract method  (20)  **Single Abstract Method** | | **7.** | unified interface to a more complex subsystem  **Facade** | | **11.** | the error thrown when a method call could cause the fields of the class to contain incorrect values  **IllegalStateException** | | **12.** | special value that means that a reference does not refer to anything  **NULL** | | **14.** | mechanism used to avoid code reuse in related classes  **abstract class** | | **16.** | program that is used demonstrate how a model should be used  **driver** | | **17.** | way to test if two values are not equals  **instanceof**  \_ \_ s\_ \_ \_n\_ \_e \_ \_ a\_ \_ | | **19.** | a general collection that can store primitive types  **array** | | **21.** | the determination of which method to call based on the type of the calling object  **dynamic dispatch** | | **22.** | testing that is written without knowing anything about the implementation  **blackbox** | | **23.** | another word for a parent class (10)  **superclass** | | **27.** | the type of the parameter passed to the `map` method  **function** | | **28.** | the component of the MVC pattern responsible for handling user input  **Controller** | | **30.** | method used to generate a representation of an object that can be printed to the console  **toString** | | **31.** | the need to declare variables before using them  **Static data types** | | **32.** | testing that contains specific knowledge of the implementation  **whitebox** | | **33.** | A = A  **Reflexivity** | | **34.** | the component of the MVC pattern responsible for showing results to the client  **View** | | **36.** | the SOLID principle that requires backwards compatibility  **Liskov's substitution** | | **37.** | declaring variables of an interface type instead of the concrete type  **Dependency inversion** | | **38.** | to pass responsibility to another another object used as a field  **disposable** | |  | |  |  | | --- | --- | | **Down** | | | **1.** | the type of the parameter passed to the reduce method  **bifunction** | | **2.** | each class should represent one thing in our design  **SingleResponsibility** | | **3.** | if A = B then B = A  **Symmetry** | | **6.** | the error thrown when a client passes in incorrect values in a method call  **IllegalArgumentException** | | **8.** | degree to which two classes depend on each other  **Coupling** | | **9.** | placing data and behaviors into a single component  **Encapsulation** | | **10.** | true or false  **boolean** | | **13.** | the component of the MVC pattern responsible for the functionality of the application  **model** | | **15.** | when a parameter is named the same thing as a field  **static** | | **18.** | another word for a child class  **subclass** | | **20.** | representing essential features without implementation details  **abstraction** | | **24.** | using private as much as possible  **Secure programming** | | **25.** | another word for inheritance relationships (14)  **super-sub class** | | **26.** | the type of the parameter passed to the `filter` method  **Predicate** | | **28.** | method used to determine the absolute ordering between two instances  **compareTo** | | **29.** | how elements are organized in software and how they relate to each other  **architecture** | | **35.** | degree to which a class has a well-defined purpose  **coupling** | |
|  |  |  |