Content Objective: Students will implement subclasses that inherit and override superclass’s in Java applications.

|  |  |
| --- | --- |
| **On the Tech Horizon (10pts.)**  **link to a tech/coding related article or journal no more than one month old (no blogs or reddit clones see below)** | |
| URL: |  |
| Reaction/Commentary: |  |

|  |  |
| --- | --- |
| **Tech Terms and History (20pts.)**  **vocabulary from BJ p.421-462 and The Information Chapter 8 (definition/commentary/significance in your words)** | |
| Inheritance | A relationship between larger classes (superclass) and smaller, more specific classes (subclass) |
| Superclass | The larger, more generalized class at the top of the class tree |
| Subclass | The smaller, more specialized class which inherits from the superclass |
| Substitution principle | Any object from a subclass can be used in place of a superclass |
| Summarize “Programming tip 9.1” | If a difference in two objects can be brought out my a simple difference in variables, then they shouldn’t be classified as separate subclasses |
| Overriding | Any methods in a subclass that exists in a superclass, the method is overrided |
| Syntax to access superclass method | Use super.method(); from the subclass |
| Overloading | Overloading is when there are two methods with an identical name, but with different parameters |
| Explain difference between overloading and overriding | Overloading is compile time, overriding is runtime. Overloading can be done in the same class while overriding is with multiple subclasses |
| Syntax for calling superclass constructor | Public Class(parameters){  super(arguments); //must be placed as the first line inside of a constructor  } |
| Summarize the goal of polymorphism | BJ p.437-438 |
| Dynamic method lookup | BJ p.438 |
| Program Extensibility | BJ p.438 |
| Implicit Parameters | BJ p.440 |
| Abstract Classes | BJ p.441 |
| Abstract methods | BJ p.441 |
| Concrete class | BJ p.441 |
| Final Access Modifier | BJ p.442 |
| Protected Modifier | BJ p.442 |
| Classes declared without “extends” | BJ p.448 |
| Hash code | BJ p.449 |
| Syntax to override “toString” method | BJ p.449 |
| Best practice for comparing objects | BJ p.450 |
| Summarize the “instanceof” operator | BJ p.451 |
| Describe the tension between Wiener and Shannon… | TI p.234 |
| Describe the proposed science of “cybernetics” | TI p.235-43 |
| What is “discreet chaos”? | TI p.237 |
| How is feedback “information”? | TI p.238 |
| What was the primary function of the ENIAC? | TI p.239 |
| List the technologies in the ENIAC, Mark I, and Colosus. | TI p. 239 |
| What are the two classifications of computing machines – how do they differ? | TI p. 244 |
| What is the measurement of transmitted information? | TI p. 246 |
| Describe how Shannon’s maze added to binary logic. | TI p. 250 |

|  |  |
| --- | --- |
| **Code Snippets (30pts.)**  **only submit snippets or classes no full programs required (test and run in IDE, then copy/paste applicable code frag)** | |
| E9.3, E9.6-7 |  |
| E9.8 |  |
| E9.9 |  |
| E9.10-13 |  |

|  |  |
| --- | --- |
| **Code Challenge (30pts.)**  **full functioning application sent to** [**GitHub**](https://github.com/SkylineHigh/CSAdvanced/tree/master/10%20Overriding) | |
| You may choose any of the following code challenges: P9.1-6 or provide evidence of inheritance (superclass and subclasses) in your group project application (specifically using overriding and/or polymorphism.) | |
| Notes: | https://github.com/SkylineHigh/CSAdvanced/tree/master/CHAPTER-9-CODE-CHALLENGE/kris |

|  |  |
| --- | --- |
| **Badge Progress (10pts.)**  **building your coding profile: Java coding training site to earn badges (recommended site** [**http://coderbyte.com**](http://coderbyte.com) **)** | |
| Screenshot/URL: |  |
| Notes/Issues: |  |

|  |  |
| --- | --- |
| **Notes**  **your notes** | |
| Notes: |  |