|  |  |  |
|---|---|---|
|  | **Team #:** | 39 |
|  | **Marked by:** | Dr. Barcomb |
|  | **Mark:** | 90 |
|  |  |  |
| **Measure** | **Evaluation** | **Comments** |
| Static is used where appropriate / is not used where inappropriate | Somewhat | NUTRITIONAL_NEEDS is static, but CLIENT_ID could also be static since each client category is its own class |
| Final is used for immutable values | Somewhat | NUTRITIONAL_NEEDS, CLIENT_ID, ITEM_ID are final, but FoodItem nutrition won't change either |
| Number of arrows which are inconsistent (type, direction, cardinality) with relationships shown in classes | 0 |  |
| Number of relationships which do not make sense or are inconsistent with specifications | 0 |  |
| Distinction is made between "hamper" level and "order" level | Somewhat | The variable family in Request does not appear to allow for the possibility that there will be hampers associated with multiple families in the same request. There is 1 hamper per family, but there could be multiple hampers per order - see discussion board post. |
| All other specifications appear to be included in the design | Yes |  |
| Number of UML notation errors (return types, parameters, visibility, capitalization, throwing exceptions associated with particular method, etc) | 0 | Depiction of attribute types is different than approach used in class, but is also correct |
| Diagram contains meaningful, testable public methods which perform actual work | Yes |  |
| Number of method, variable and classnames which are vague or do not describe intended use (scope) correctly | 0 |  |
| assumed) | Incorrect hardcoding | Use of hardcoded defaults for NUTRITIONAL_NEEDS; these values should come from the database |
| Diagram is tidy, legible (sufficient space between classes, minimal lines crossing, etc) | Somewhat easy to read | Line breaks in longer lines (e.g., AdultFemale constructor) can lead to some confusion; please make classes wider to accommodate |
| There is at least one appropriate use of inheritance and/or interface | Yes |  |
| There is at least one appropriate use of aggregation and/or composition | Yes |  |
| Validity of input is checked (exceptions may be thrown) | Yes | IllegalArgumentException is thrown, but also consider that writing to a file can cause exceptions (which you will probably want to catch at the user interface level) |
| Design is object-oriented and practices encapsulation | Yes | Good overall design; interesting use of a Nutrition class to store common information, nice use of abstract class |
| Number of missing constructors, getters, and setters (setters are not required for final, constructors are not required for static, getters are not required for public) | 4 | No getters for CLIENT_ID |
|  |  |  |
| **Additional comments:** | **Reminders for TA:** | GUI and database connections do not need to be depicted, they can be "magic" |
| You do not need to capture age, gender, and mobility for a person (or hamper), although we do not mind if you expand upon the specifications (as long as original functionality is still present). |  | Terminal input is not required |
| Overall a good diagram, demonstrating an understanding of OOP design! Some misunderstanding of specifications which should be resolved. |  | Client types (adult female, etc), client IDs, and database fields (protein, calories, etc) can be assumed. |