Assignment #2

DDR, Spring 2024

TASK (A): Design Principle: Encapsulate that varies

Encapsulate What Varies, or 'Encapsulate What Changes' is the technique of reducing the impact of frequently changing code by encapsulating it. Encapsulating what varies is a technique that helps us handle frequently changing details. Code tends to get tangled when it is continuously modified due to new features or requirements. By isolating parts which are prone to change we limit surface area that will be affected by a shift in requirements.

Example 1

```
// ✗ This is hard to understand and
subject to change.
// We may need to check if a book is
reserved.
function checkoutBook(customer, book) {
  if (
    customer &&
    customer.fine <= 0.0 &&
    customer.card &&
    customer.card.expiration === null &&
    book &&
    !book.isCheckedOut
  ) {
    customer.books.push(book)
    book.isCheckedOut = true
  }
  return customer
}
```

```
// ✓ This is easy to read and won't change even if the
checkout requirements vary.
function checkoutBook(customer, book) {
 if (customer.canCheckout(book)) {
  customer.checkout(book)
 }
 return customer
}
```

Example 2

```
if (pet.type() == dog) {
  pet.bark();
} else if (pet.type() == cat) {
  pet.meow();
} else if (pet.type() == duck) {
  pet.quack()
}
```

```
or you can write code that looks like this:
pet.speak();
```

Now create two applications in **java** with and without "Encapsulate that varies" principle.

TASK (B): Create small sample applications in **java** demonstrating Abstract factory pattern.

For both Task (a) and Task (b) given above, document your scenario in depth in a textual form and show screenshots or output and UML diagrams (class and interaction etc.). Do not copy paste examples from any public domain or internet? Create your own application.

Note:

1. Assignment must be submitted on Google forms (MS word Assignment report).
2. Last date of submission is **26-02-2024 11:00 AM (sharp)**
3. Assignments will not be accepted after due date.
4. Only 1 submission is allowed, if you once submit it then you are allowed to update your submission.
5. You need to submit in your section only. If you submit in wrong section or both sections or if you violate any assignment instructions then expect negative marking.
6. Email submission not allowed. Emails queries asking hints for solution will result in negative marking.
7. If you upload empty or corrupted archive, you will get zero marks. Hence double check before uploading.
8. Plagiarism, if detected, will result in zero marks.
9. Do not use the sample shown in the reading assignment / text books or form internet.
10. Upload a Zip archive (name is as Ass02_name_roll_number_section.ZIP) having following contents:
11. MS word document in /Doc directory. All UML diagrams in /UML directory
12. Contents of the MS Word report:
    a. Cover Page of Assignment must contain: Student name, Roll no, Date of submission.
    b. Attach screenshot of the question paper after cover page.

c. Solutions to all the questions, figures, source code, output screenshots and textual descriptions.

13. Source code in /code directory