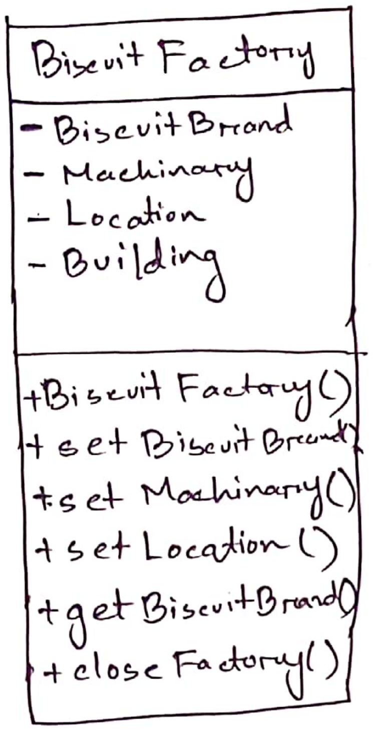
17050 45

Page-1

① We can apply the renovation scenario in a maintaining the codes in different classes. We may think we have a 'Factory' class, which has functionality like produceBiscuits() and others.

It can be

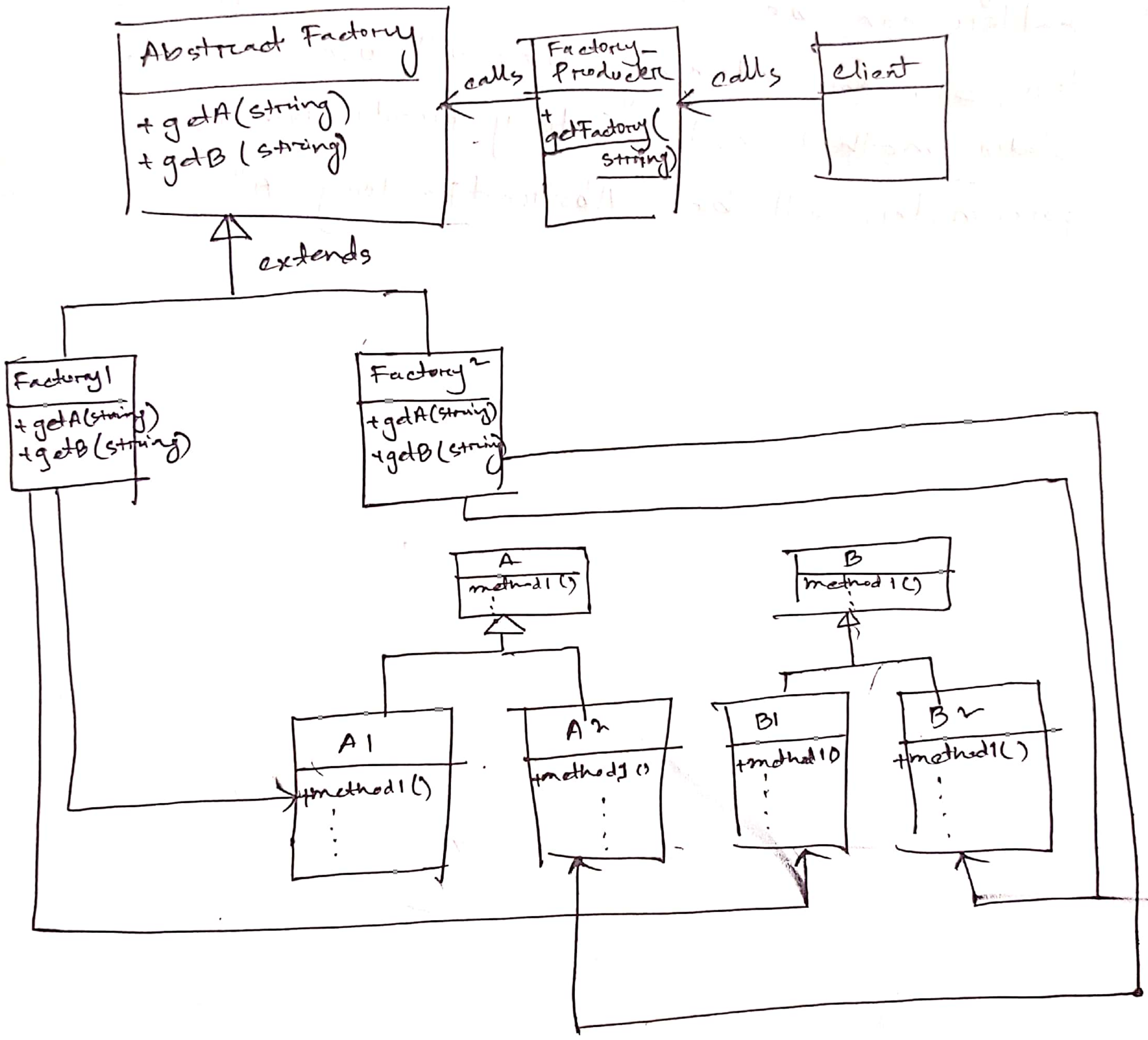


Now, in our code for renovation process, we may renovate oldFactory with getter and setter methods.

We can do something like:—

```
BiscuitFactory renovate (Biscuit Factory old_Factory, String new_Machinary,
                                                     String Location) {
BiscuitFactory newFactory = old_Factory;
        new_Factory. set Machinary(new Machinary)
        new_Factory. set Locatin(new_Location);
        old_Factory. close Factory();
        return new_Factory;
}
```

② Simplified diagram-



-AbstractFactory pattern can index the factory classes. whenever we need an indexing of factory classes, ~~ast~~ abstract factory pattern is best place to put it. Inside factory producer, we know all the names and types of ~~fact~~ all factories. That's why, it can be easily done there. Along side being a factory for factory, abstract factory pattern can be used to index the factories too. So, for indexing, we will use another static method inside factory producer. Its parameter will be "AbstractFactory A".