CAMPUS 02 GRAZ
FACHHOCHSCHULE DER WIRTSCHAFT

Fachhochschul-Akademischer Lehrgang „Software Engineering“

Klausur aus der Lehrveranstaltung

„SOFTWARE DESIGN“

Hofer Christian

Nachklausur am 17.06.2020
2.Termin

2. SEMESTER

WS 19/20

Name: _____

Beurteilung: _____

Klausurdauer: 120 Minuten

Sie dürfen alle Ihre Unterlagen, alte Programme und Internetquellen verwenden. Kommunikation oder Austausch in jedweder Art (analog oder digital) ist nicht erlaubt.

Über Moodle haben Sie Zugriff auf ihr Startpaket. Bitte geben Sie am Ende der Zeit Ihr Projektverzeichnis als ZIP Datei über Moodle ab.

Die Junit Tests sind nur zur Hilfe und Orientierung für Sie und garantieren keine vollständige Korrektheit Ihres Codes. (Achtung ist auskommentiert…)

Beispiel 1 Nokia (50 %)

Nokia besinnt sich wieder auf seine Wurzeln und will neben Netzwerkausstattung auch wieder Gummistiefel herstellen. Wenden Sie das **Factory Pattern** an um in zwei Fabriken Handy Basisstationen bzw. Stiefel (in Universalgröße) zu produzieren.

Die **Product** Klasse ist die Ausgangsbasis für alle Produkte und ist bereits großteils umgesetzt. Fügen Sie dieser Klasse noch eine abstrakte Methode *getSpeed()* hinzu die einen String als Rückgabewert liefert.

Implementieren Sie die darauf basierenden Produktklassen, welche als Speed den String in der Klammer zurückliefern:

- Boots („by foot“)
- BaseStation3G („3G“)
- BaseStation4G („4G“)
- BaseStation5G („5G“)

ProductType ist ein enum Typ zur Unterscheidung der Produkte.

Die **NokiaFactory** gibt mit seiner *getProduct* Methode (Methodensignatur nicht ändern!) den firmenübergreifenden Produktionsprozess vor, der vorsieht, dass jedes Produkt eine Seriennummer erhält. Adaptieren Sie diese Klasse, damit Sie die Ausgangsbasis für zwei Fabriken ist.

Die **BootFactory** kann nur Boots Objekte erstellen und liefert sonst null retour.

Die **BaseStationFactory** kann die drei Basestations erstellen und liefert bei einer anderen Produktauswahl null retour.

Erstellen Sie ein eigenständiges **Main** und testen Sie ob beide Fabriken nur die korrekten Produkte erzeugen.

Beispiel 2 Spiel (50 %)

Sie erstellen ein innovatives Rollenspiel für Konsolen und Smartphones, welches je nachdem wann Sie spielen Sie mit zufälligen dynamischen Ereignissen überrascht. Mittels **Observer Pattern** möchten Sie an alle aktiven SpielerInnen die neuen zufälligen Spieleevents übertragen. Verwenden Sie für Ihr Observer Pattern die Interfaces **GameClient** und **GameServer**.

Das **GameClient** Interface ist bereits umgesetzt und darf nicht verändert werden. Die *update* Methode überliefert die neuen Events welche über die Klasse **EventData** kommuniziert werden. Sie ist bereits umgesetzt und enthält einen Link auf die neuen Spieldaten und eine Kurzbeschreibung. Weiters finden sich im **GameClient** noch die Methode *startGame* – erst

wenn diese aufgerufen wird soll sich ein konkreter Client beim Subject registrieren bzw. eine Referenz auf das Subject speichern. Nach Aufruf der *endGame* Methode soll sich ein Client abmelden und die Referenz auf null setzen.

Die Klasse **AppClient** soll einen Smartphone Client für das Rollenspiel umsetzen. Wenn der AppClient ein neues Event bekommt dann soll das nur behandelt werden, wenn zumindest 20% Akku über sind und die Version des Spiels mindestens 1000 ist: Den Beschreibungstext auf Konsole ausgeben und die Adresse auf *lastEventURL* sichern. Falls die Bedingung nicht zutrifft nichts tun.

ConsoleClient steht für eine Spielkonsole und soll ebenso einen Spielclient umsetzen. Wenn ein **ConsoleClient** ein neues Event bekommt, soll es nur behandelt werden, wenn der Systemname auf *system* ungleich „Nintendo“ ist: Bei Behandlung soll ebenso die Beschreibung ausgegeben werden und die Eventadresse auf *lastEventURL* gespeichert werden.

Der **RoleplayServer** ist die zentrale Steuereinheit für alle Spiele. Über ihn können neue Spielevents an alle registrierten Spieleclients geschickt werden.

Implementieren Sie die *newEvents* Methode: Erzeugen Sie ein neues *EventData* Objekt, merken Sie sich dieses auf der Referenz *lastEvent* und informieren Sie alle Spieleclients über eine eigene Methode zum Benachrichtigen aller Clients. (= Das Informieren nicht direkt in *newEvents* Methode umsetzen)