Celkový popis

Semestrální práce představuje komplexní simulaci chytrého domu, která zahrnuje interakci živých bytostí s chytrými zařízeními, stejně jako řízení chování a strategií těchto bytostí. Simulace je obohacena dynamickým počasím a pokročilými funkcemi řízení zařízení.

Řízení událostí:

- *Události bez zařízení:* V simulaci jsou realizovány události, které nevyžadují interakci s zařízeními.

- *Události se zařízeními:* Různé bytosti (dospělí, děti) interagují s zařízeními různě, využívajíce pattern "Visitor".

Dynamické počasí:

- Systém počasí se mění od slunečného po deštivé nebo oblačné, ovlivňuje teplotu a intenzitu světla.

- Změna počasí od 4 ráno do 4 odpoledne: Od 4 ráno do 4 odpoledne dochází ke změně teploty a intenzity světla. Od 4 ráno začíná oteplovat a světlit, vrchol dosahuje v 4 odpoledne.(Teplota a intenzivita světla se mění nelinearní pomocí algoritmu s počítaním faktoru, posle přidá na konec náhodnou intenzitu nebo teplotu, tj. -1, 0, nebo 1)

Řízení zařízení a jejich stavů:

- *Řízení závěsů a topení:* Device “Curtain Manager” reguluje polohu závěsů v závislosti na intenzitě světla, zatímco device “Temperature Manager” nastavuje teplotu v domě podle vnější teploty.

- *Dekorátory pro zařízení:* Zařízení mohou vyvolávat různé situace, jako jsou znečištění, poruchy, požáry a záplavy. V případě vzniku požáru nebo záplavy je nutná oprava zařízení. Například, pokud počítač způsobí požár, také se rozbije, což vyžaduje následnou opravu.

Smart panel a senzory:

- *Senzory:* V domě jsou umístěny různé senzory: smoke sensor, flood sensor, , které zajišťují automatické řízení a reakci na změny vnějšího prostředí.

- *Smart Panel*: V domě je nainstalován centrální chytrý panel (Smart Panel), který řídí různá zařízení a systémy pomocí patternu „Observer“.

- *Strategie chování:* Sensory detekují emergency situace, na dalším stepu simalace SmartPanel uvědomí Alive Entity a devicy. Bytosti se chovají různě v závislosti na aktuální strategii. Například při požáru se děti uklidňují a čekají, až požár skončí, zatímco rodiče hasí.

Řetězec a obnova událostí:

- Řetězec událostí:
  - 1) Při poruše zařízení bytost provádí sérii událostí pro jeho opravu.
  - 2) Bytost může jít do obchodu pro nákupy, pokud v domě dojde jídlo.
  - 3) Bytost si může vzít jídlo z ledničky, a pokud je do simulace přidána mikrovlnná trouba, půjde k ní, pokud je v jiné místnosti, a ohřeje si ho.
- Obnova událostí po změně strategií: Přerušené události se ukládají a obnovují se po změně strategií chování bytostí.

- Přesun bytostí mezi místnostmi: Bytosti se mohou pohybovat po domě pro interakci s zařízeními v jiných místnostech.

Důležité poznámky

Částečná validace JSON konfigurace: Validace konfigurace JSON souboru je realizována pouze částečně. Pozor: 1) Json file “configuration_big” obsahuje všechny možné devicy.

2) Fridge musí být nastavlen.

3) V jedné místnosti může být pouze jeden "Heater".

4) Curtain nemusíte instalovat, automaticky se nainstaluje do každého okna.

5) Auto se instaluje automaticky.

Použité patterny:

1.Singleton

- Cíl: Zajistit vytvoření pouze jedné instance třídy a poskytnout globální přístupový bod k této instanci.
- Třídy: TimeManager, House, Weather, AdultStrategyFactory.
- Příklad: Ve třídě TimeManager se používá soukromý konstruktor a statická metoda getInstance() pro vytvoření a správu jediné instance.

2.Factory Method

- Cíl: Definovat rozhraní pro vytváření objektu, ale umožnit podtřídám měnit typ vytvářených objektů.
- Třídy: House, TimeManager, AdultStrategyFactory, AnimalStrategyFactory, ChildStrategyFactory.
- Příklad: Metody getDefaultStrategy() v těchto třídách jsou továrními metodami, které vytvářejí různé strategie.

3.Decorator

- Cíl: Dynamicky přidávat nové povinnosti objektům bez změny jejich struktury.
- Třídy: FlammableDevice, FloodingDevice, MessCausingDevice.
- Příklad: Tyto třídy rozšiřují funkčnost BreakableDevice, přidávají nové charakteristiky (například schopnost způsobit požár).

4.Observer

- Cíl: Definovat závislost typu "jeden k mnoha" mezi objekty tak, že při změně jednoho objektu jsou všechny od něj závislé objekty upozorněny a automaticky aktualizovány.
- Třídy: TimeManager (jako vydavatel) a různé entity (například Adult, Child, Animal jako předplatitelé).
- Příklad: TimeManager upozorňuje přihlášené entity na každý krok simulace.

5.Visitor

- Cíl: Umožnit přidávání nových operací k objektům bez změny tříd těchto objektů.
- Třídy: HumanEntity a jeho podtřídy (Adult, Child).
- Příklad: Metody visit... v HumanEntity umožňují zpracovávat různé typy zařízení.

6.Strategy

- Cíl: Definovat rodinu algoritmů, zapouzdřit každý z nich a učinit je vzájemně zaměnitelnými. Strategie umožňuje měnit algoritmus nezávisle na klientech, kteří jej používají.
- Třídy: AdultDefaultStrategy, AnimalDefaultStrategy, ChildDefaultStrategy.
- Příklad: Tyto třídy představují různé strategie generování událostí pro různé typy entit.

7.State

- Cíl: Umožnit objektu měnit své chování, když se změní jeho vnitřní stav.
- Třídy: Weather, SunnyState, RainyState, CloudyState.
- Příklad: Weather mění své chování v závislosti na aktuálním stavu počasí (SunnyState, RainyState, CloudyState, které vrátí teplotu a intenzivitu světla venku).

8.Stream

- Cíl: Zjednodušit práci s kolekcemi a poli dat s využitím funkčního přístupu.
- Třídy: House.
- Příklad: V metodě getAllDevices() třídy House se používají proudy pro zpracování seznamů zařízení a oken, getToilets().

9.Functional Interface (@FunctionalInterface)

- Cíl: Definovat rozhraní určené pro implementaci v lambda výrazech a metodách.
- Třídy: Event.EndFunction.
- Příklad: EndFunction v Event se používá k definování operací, které se provádějí na konci události.

10.Lazy Initialization

- Cíl: Odložená inicializace se používá ke snížení nákladů na zdroje tím, že se objekt inicializuje pouze tehdy, když je poprvé potřeba.
- Třídy: AdultStrategyFactory, AnimalStrategyFactory, ChildStrategyFactory.
- Příklad: V těchto třídách strategií se používá odložená inicializace pro vytváření instancí strategií (defaultStrategy, fireStrategy, floodStrategy).

11.Template Method

- Cíl: Vzor "Šablonová metoda" definuje kostru algoritmu v metodě, odkládá některé kroky algoritmu do podtříd. Vzor umožňuje podtřídám přepisovat určité kroky algoritmu bez změny jeho struktury.
- Třídy: Device, AliveEntity (POUZE REALIZUJE MYŠLENKU PATTERNU) (a jeho podtřídy, jako jsou Adult, Child, Animal).
- Příklady: Ve třídě Device a jejích podtřídách:
    - Metoda consumeEnergy() v Device definuje obecný algoritmus spotřeby energie, používá krok času a standardní spotřebu energie.
        - Abstraktní metoda getDefaultConsumptionInMinute() vyžaduje od podtříd definovat konkrétní spotřebu energie, což umožňuje různým zařízením mít své jedinečné hodnoty spotřeby.

- Ve třídě AliveEntity a jejích podtřídách (REALIZUJE JENOM MYSLENKU PATTERNU):
  - Metoda doStep(int context) v AliveEntity definuje obecný algoritmus provádění kroku simulace, včetně generování událostí a možné změny místnosti.
  - Metoda generate() v EventGeneratingStrategy se používá pro generování konkrétních událostí, což umožňuje různým entitám (dospělým, dětem, zvířatům) mít své jedinečné strategie generování událostí.

12.Builder

- Cíl: Umožnit vytváření složitého objektu postupně a poskytnout možnost použít různá zobrazení objektu.
- Třídy: Event.EventBuilder.
- Příklad: Ve třídě Event se používá vnitřní třída EventBuilder pro postupné vytváření složitých objektů událostí.