1. **KASUS(100%)**
2. **Anda diminta untuk membuat design pattern perangkat lunak Parkir dengan asumsi sebagai berikut:**

* Tempat parkir memiliki beberapa tingkatan dan memiliki beberapa baris titik.
* Tempat parkir dapat memarkir sepeda motor, mobil, dan bus.
* Tempat parkir terdapat spot motor, spot padat, dan spot besar.
* Sepeda motor bisa parkir di sembarang tempat.
* Sebuah mobil dapat parkir di satu tempat kecil atau satu tempat besar.
* Sebuah bus dapat parkir di lima tempat besar yang berurutan dan dalam baris yang sama. Tidak bisa parkir di tempat kecil.

Dalam implementasi di bawah ini, kami telah membuat Kendaraan kelas abstrak, yang mewarisi mobil, bus, dan sepeda motor. Untuk menangani ukuran tempat parkir yang berbeda, kami hanya memiliki satu kelas ParkingSpot yang memiliki variabel anggota yang menunjukkan ukuran.

1. **[LO 1, 30 poin]** Berdasarkan soal no 1, analisa dan gambarkan implementasi design pattern berikut:

* Factory Method
* Prototype
* Adapter

**Jawaban:**

**Vehicle.java**

**import** java.util.ArrayList;

**public** **abstract** **class** Vehicle {

**protected** ArrayList<ParkingSlot> parkingSlots = **new** ArrayList<ParkingSlot>();

**protected** String licPlate, companyName;

**protected** **int** slotsNeeded;

**protected** VSize siz;

**public** **int** getSlotsNeeded() {

**return** slotsNeeded;

}

**public** **void** parkInSlot(ParkingSlot s) {

parkingSlots.add(s);

}

**public** VSize getSize() {

**return** siz;

}

**public** **void** clearSlots() {

**for** (**int** i = 0; i < parkingSlots.size(); i++) {

parkingSlots.get(i).removeVehicle();

}

parkingSlots.clear();

}

**public** **abstract** **boolean** canFitInSlot(ParkingSlot spot);

}

**VSize.java**

**public** **enum** VSize {

***CarSize***, ***Motorcycle***, ***Buses***,

}

**Motorcycle.java**

**public** **class** Motorcycle **extends** Vehicle{

**public** Motorcycle(String licPlate, String companyName) {

slotsNeeded = 1;

siz = VSize.***Motorcycle***;

**this**.licPlate = licPlate;

**this**.companyName = companyName;

}

@Override

**public** **boolean** canFitInSlot(ParkingSlot spot) {

**return** spot.getSize() == VSize.***CarSize*** || spot.getSize() == VSize.***Motorcycle***;

}

}

**Car.java**

**public** **class** Car **extends** Vehicle{

**public** Car(String licPlate, String companyName) {

slotsNeeded = 1;

siz = VSize.***CarSize***;

**this**.licPlate = licPlate;

**this**.companyName = companyName;

}

@Override

**public** **boolean** canFitInSlot(ParkingSlot spot) {

**return** spot.getSize() == VSize.***CarSize***;

}

}

**Buses.java**

**public** **class** Buses **extends** Vehicle{

**public** Buses(String licPlate, String companyName) {

slotsNeeded = 1;

siz = VSize.***Buses***;

**this**.licPlate = licPlate;

**this**.companyName = companyName;

}

@Override

**public** **boolean** canFitInSlot(ParkingSlot spot) {

**return** spot.getSize() == VSize.***Buses***;

}

}

**Level.java**

**public** **class** Level {

**private** **int** floor;

**private** ParkingSlot[] slots;

**private** **int** availableSlots = 0;

**private** **static** **final** **int** ***SLOT\_PER\_ROW*** = 10; //Per Baris Muat 10

**public** Level(**int** flr, **int** numberSlots) {

floor = flr;

availableSlots = numberSlots;

slots = **new** ParkingSlot[numberSlots];

**int** largeSlots = numberSlots / 4;

**int** bikeSlots = numberSlots / 4;

**int** compactSlots = numberSlots - largeSlots - bikeSlots;

**for** (**int** j = 0; j < numberSlots; j++) {

VSize siz = VSize.***Motorcycle***;

**if** (j < largeSlots + compactSlots) {

siz = VSize.***CarSize***;

} **else** {

siz = VSize.***Buses***;

}

**int** row = j / ***SLOT\_PER\_ROW***;

slots[j] = **new** ParkingSlot(**this**, row, j, siz);

}

}

**public** **int** availableSlots(){

**return** availableSlots;

}

**public** **boolean** parkVehicle(Vehicle vh) {

//Function buat Mencari tempat parkir buat kendaraan

**if** (availableSlots() < vh.getSlotsNeeded())

**return** **false**;

**int** slotNumber = findAvailableSlots(vh);

**if** (slotNumber < 0)

**return** **false**;

System.***out***.print(", Nomor Slot " + slotNumber);

**return** parkStartingAtSlot(slotNumber, vh);

}

**private** **boolean** parkStartingAtSlot(**int** num, Vehicle vh) {

// Update Slot yang tersedia

vh.clearSlots();

**boolean** success = **true**;

**for** (**int** j = num; j < num + vh.slotsNeeded; j++) {

success &= slots[j].park(vh);

}

availableSlots = availableSlots - vh.slotsNeeded;

**return** success;

}

**private** **int** findAvailableSlots(Vehicle vh) {

**int** slotsNeeded = vh.getSlotsNeeded();

**int** lastRow = -1;

**int** slotsFound = 0;

**for** (**int** j = 0; j < slots.length; j++) {

ParkingSlot spot = slots[j];

**if** (lastRow != slots[j].getLane()) {

slotsFound = 0;

lastRow = slots[j].getLane();

}

**if** (slots[j].canFitVehicle(vh)) {

slotsFound = slotsFound + 1;

} **else** {

slotsFound = 0;

}

**if** (slotsFound == slotsNeeded) {

**if**(vh.siz == VSize.***CarSize***) {

System.***out***.print("Sebuah Mobil Diparkir di "); }

**else** **if**(vh.siz == VSize.***Motorcycle***) {

System.***out***.print("Sebuah Motor Diparkir di ");}

**else{**

System.***out***.print("Sebuah Bus Diparkir di ");}

System.***out***.print("Baris Nomor " + lastRow);

**return** j - (slotsNeeded - 1);

}

}

**return** -1;

}

// Ketika ada kendaraan yang keluar gedung

// Update Jumlah Slot Parkir

**public** **void** slotFreed() {

availableSlots = availableSlots + 1;

System.***out***.println("Slot yang masih tersedia di Level ini :" + availableSlots);

}

}

**ParkingSlot.java**

**public** **class** ParkingSlot {

**private** Vehicle vh;

**private** VSize siz;

**private** **int** lane;

**private** **int** slotNumber;

**private** Level l;

**public** ParkingSlot(Level lvl, **int** r, **int** num, VSize vs) {

l = lvl;

lane = r;

slotNumber = num;

siz = vs;

}

**public** **boolean** isAvailable() {

**return** vh == **null**;

}

**public** **boolean** canFitVehicle(Vehicle vh) {

**return** isAvailable() && vh.canFitInSlot(**this**);

}

**public** **boolean** park(Vehicle vObj) {

**if** (!canFitVehicle(vObj)) {

**return** **false**;

}

vh = vObj;

vh.parkInSlot(**this**);

**return** **true**;

}

**public** **int** getLane() {

**return** lane;

}

**public** **int** getSlotNumber() {

**return** slotNumber;

}

**public** VSize getSize() {

**return** siz;

}

**public** **void** removeVehicle() {

l.slotFreed();

vh = **null**;

}

}

**ParkingSpot.java**

**import** java.util.ArrayList;

**import** java.util.\*;

**import** java.util.HashMap;

**public** **class** ParkingSpot{

**private** Level[] lvls;

**private** **final** **int** NUM\_LVLS = 2;

**private** **int** numberSlots;

Map<String, ArrayList<String>> CompanytoVeh;

**public** ParkingSpot(**int** numberSlots) {

**this**.numberSlots = numberSlots;

**this**.CompanytoVeh = **new** HashMap<String, ArrayList<String>>();

lvls = **new** Level[NUM\_LVLS];

**for** (**int** j = 0; j < NUM\_LVLS; j++) {

lvls[j] = **new** Level(j, numberSlots);

System.***out***.println("Level " + j + " dibuat sejumlah " + numberSlots + " " + "slots");

}

}

// Method Meyimpan Data Kendaraan yang diparkir seperti Nama Pemilik dan Nomor Kendaraan

**public** **boolean** parkVehicle(Vehicle vh) {

System.***out***.println(" ------------------------------------ ");

**for** (**int** i = 0; i < lvls.length; i++) {

**if** (lvls[i].parkVehicle(vh)) {

System.***out***.println(" Level " + i + " Dengan Plat Nomor " + vh.licPlate + " dari " + vh.companyName);

**if** (**this**.CompanytoVeh.containsKey(vh.companyName)) {

ArrayList<String> regNoList = **this**.CompanytoVeh.get(vh.companyName);

**this**.CompanytoVeh.remove(vh.companyName);

regNoList.add(vh.licPlate);

**this**.CompanytoVeh.put(vh.companyName, regNoList);

} **else** {

ArrayList<String> regNoList = **new** ArrayList<String>();

regNoList.add(vh.licPlate);

**this**.CompanytoVeh.put(vh.companyName, regNoList);

}

**return** **true**;

}

}

System.***out***.println("TEMPAT PARKIR SUDAH PENUH");

**return** **false**;

}

// Method ketika Kendaraan Keluar (Hapus dari List)

**public** **boolean** leave(Vehicle vh, **int** lvl) {

System.***out***.println(" ------------------------------------------ ");

lvls[lvl].slotFreed();

System.***out***.println("Slot freed from Level " + lvl + " and exited " + vh.licPlate + " of " + vh.companyName);

ArrayList<String> vhList = **this**.CompanytoVeh.get(vh.companyName);

// Untuk Cek apakah kendaraan yang keluar ada di List

**if** (vhList.contains(vh.licPlate)) {

vhList.remove(vh.licPlate);

}

**return** **true**;

}

// Method untuk Print kendaraan di List.

**public** **void** ComapnyParked(String companyName) {

System.***out***.println(" ------------------------------------------ ");

ArrayList<String> vhList = **this**.CompanytoVeh.get(companyName);

System.***out***.print("Kendaraan dari " + companyName + " : ");

**for**(String vl : vhList) {

System.***out***.print(vl + "\t");

}

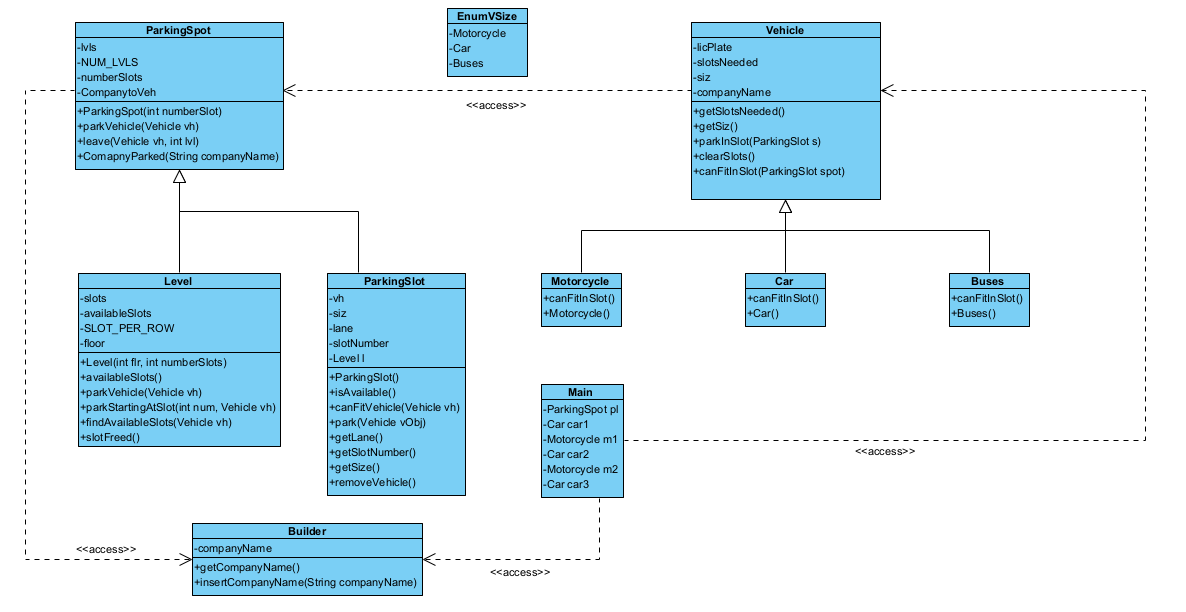
System.***out***.println();

}

}

1. **[LO 2, 10 poin]** Berdasarkan soal no.1, gambarkan kelas diagram dari sistem yang ada. Kelas diagram yang dibuat sudah menggabungkan design pattern yang diterapkan pada soal no. 1!

**Jawaban:**



1. **[LO 3, 20 poin]** Buatlah builder untuk class ParkingSpot. Contohkan penggunaan builder Anda di class Main.

**Jawaban:**

**Builder.java**

**public** **class** Builder{

String companyName;

**public** String getCompanyName() {

**return** companyName;

}

**public** String insertCompanyName(String companyName) {

**this**.companyName = companyName;

**return** companyName;

}

}

**Main.java**

**public** **class** Main {

**public** **static** **void** main(String argvs[]) {

ParkingSpot pl = **new** ParkingSpot(2);

Car car1 = **new** Car("2301905543", "Fiona");

Motorcycle m1 = **new** Motorcycle("LJ01", "Varencia");

Car car2 = **new** Car("2021", "Tendio");

Motorcycle m2 = **new** Motorcycle("6122", "Framework");

Car car3 = **new** Car("2022", "layer");

pl.parkVehicle(car1);

pl.parkVehicle(m1);

pl.parkVehicle(car2);

Builder builder = **new** Builder();

System.***out***.println();

builder.insertCompanyName("Fiona");

pl.ComapnyParked(builder.getCompanyName());

System.***out***.println();

builder.insertCompanyName("Varencia");

pl.ComapnyParked(builder.getCompanyName());

pl.leave(m1,0);

System.***out***.println();

builder.insertCompanyName("Fiona");

pl.ComapnyParked(builder.getCompanyName());

System.***out***.println();

pl.parkVehicle(m2);

System.***out***.println();

pl.parkVehicle(car3);

}

}

1. **Anda diminta untuk merancang komputer dan diberikan asumsi sebagai berikut:**

* Komputer terbuat dari prosesor, memori dan kartu tambahan potensial (suara, dll),
* Prosesor adalah CPU atau kartu akselerator yang akan menambah perilaku ke CPU yang ada,
* Prosesor harus init (serta kartu akselerator) dan memiliki kemampuan untuk membaca / menulis dari / dalam memori, untuk memuat program ke alamat dan menjalankannya.
* Memori dapat berupa RAM atau cache,
* Cache selalu berurusan dengan RAM,
* Jika ada cache itu terutama dipertanyakan oleh prosesor,
* Komputer dapat langsung menangani hingga 3 kartu tambahan,
* Untuk menangani kartu baru, komputer harus memiliki papan ekstensi,
* Banyak kartu suara dengan perilaku berbeda ada.

1. **[LO 1, 30 poin]** Berdasarkan soal no 2., analisa dan gambarkan implementasi design pattern berikut:

* Factory Method
* Prototype
* Adapter

**Jawaban:**

**Computer.java**

**import** java.util.ArrayList;

**public** **abstract** **class** Computer {

**protected** ArrayList<ComputerSlot> computerSlots = **new** ArrayList<ComputerSlot>();

**protected** String processor;

**protected** String memory;

**protected** String additionalCard;

**public** **abstract** **boolean** canFitInSlot(ComputerSlot slot);

}

**Processor.java**

**import** java.util.ArrayList;

**public** **abstract** **class** Processor **extends** Computer{

**protected** **int** maxProgramManage = 5;

**public** **int** getMaxProgramManage() {

**return** maxProgramManage;

}

**public** **void** setMaxProgramManage(**int** maxProgramManage) {

**this**.maxProgramManage = maxProgramManage;

}

@Override

**public** **boolean** canFitInSlot(ComputerSlot slot) {

**return** **true**;

}

**public** **abstract** **void** canRead();

**public** **abstract** **void** canWrite();

**public** **abstract** **void** delProcessor(String name,ArrayList<ComputerSlot> cS);

**public** **abstract** **void** printComputerComponent(ArrayList<ComputerSlot> cS);

}

**CPU.java**

**import** java.util.ArrayList;

**public** **class** CPU **extends** Processor{

**public** CPU() {

**super**();

}

**void** addCPU(String CPUName, ArrayList<ComputerSlot> cS) {

**if**(canFitInSlot(**null**) ==**true**) {

System.***out***.println("Berhasil Insert CPU dengan Nama "+CPUName);

**if**(cS.size()==0) {

cS.add(**new** ComputerSlot(CPUName, "", "", "", "",getMaxProgramManage()+5 , 0, 1));

System.***out***.println("Maksimal Banyak Program Berjalan Menjadi "+cS.get(cS.size()-1).getMaxProgramManage());

System.***out***.println();

}**else** {

cS.add(**new** ComputerSlot(CPUName, "", "", "", "",cS.get(cS.size()-1).getMaxProgramManage()+5 , 0, 1));

System.***out***.println("Maksimal Banyak Program Berjalan Menjadi "+cS.get(cS.size()-1).getMaxProgramManage());

System.***out***.println();

}

}

}

@Override

**public** **void** canRead() {

System.***out***.println("CPU Berhasil Membaca");

}

@Override

**public** **void** canWrite() {

System.***out***.println("CPU Berhasil Menulis");

}

@Override

**public** **void** delProcessor(String name,ArrayList<ComputerSlot> cS) {

**int** count = cS.size();

**int** indeks = -1;

**for** (**int** i = 0; i < count; i++) {

**if**(cS.get(i).getCPU().equals(name)) {

indeks = i;

**break**;

}

}

**if**(indeks >= 0) {

cS.remove(indeks);

System.***out***.println("Berhasil Hapus Komponen dengan nama "+name);

}

}

@Override

**public** **void** printComputerComponent(ArrayList<ComputerSlot> cS) {

**int** jumlah = cS.size();

**if**(jumlah==0) {

System.***out***.println("Tidak ada komponen");

}**else** {

System.***out***.println("Jumlah Sisa Komponen = "+jumlah);

**for** (**int** i = 0; i <jumlah; i++) {

**if**(cS.get(i).getCheck()==1) {

System.***out***.println("Nama : "+cS.get(i).getCPU() +" (Tipe : CPU)");

}**else** **if**(cS.get(i).getCheck()==2) {

System.***out***.println("Nama : "+cS.get(i).getAccelarator() +" (Tipe : Accelerator)");

}**else** **if**(cS.get(i).getCheck()==3) {

System.***out***.println("Nama : "+cS.get(i).getRAM() +" (Tipe : RAM)");

}**else** **if**(cS.get(i).getCheck()==4) {

System.***out***.println("Nama : "+cS.get(i).getCache() +" (Tipe : Cache)");

}**else** **if**(cS.get(i).getCheck()==5) {

System.***out***.println("Nama : "+cS.get(i).getAdditionalCard() +" (Tipe : Additional Card)");

}

}

System.***out***.println();

}

}

}

**Accelerator.java**

**import** java.util.ArrayList;

**public** **class** Accelerator **extends** Processor{

**void** addAccelerator(String CPUAccelerator, ArrayList<ComputerSlot> cS) {

**if**(canFitInSlot(**null**) ==**true**) {

System.***out***.println("Berhasil Insert Accelerator dengan Nama "+CPUAccelerator);

**if**(cS.size()==0) {

cS.add(**new** ComputerSlot("", CPUAccelerator, "", "", "",getMaxProgramManage()+5 , 0, 2));

System.***out***.println("Maksimal Banyak Program Berjalan Menjadi "+cS.get(cS.size()-1).getMaxProgramManage());

System.***out***.println();

}**else** {

cS.add(**new** ComputerSlot("", CPUAccelerator, "", "", "",cS.get(cS.size()-1).getMaxProgramManage()+5 , 0, 2));

System.***out***.println("Maksimal Banyak Program Berjalan Menjadi "+cS.get(cS.size()-1).getMaxProgramManage());

System.***out***.println();

}

}

}

@Override

**public** **void** canRead() {

System.***out***.println("Accelerator Berhasil Membaca");

}

@Override

**public** **void** canWrite() {

System.***out***.println("Accelerator Berhasil Menulis");

}

@Override

**public** **void** delProcessor(String name, ArrayList<ComputerSlot> cS) {

**int** count = cS.size();

**int** indeks = -1;

**for** (**int** i = 0; i < count; i++) {

**if**(cS.get(i).getAccelarator().equals(name)) {

indeks = i;

**break**;

}

}

**if**(indeks >= 0) {

cS.remove(indeks);

System.***out***.println("Berhasil Hapus Komponen dengan nama "+name);

}

}

@Override

**public** **void** printComputerComponent(ArrayList<ComputerSlot> cS) {

**int** jumlah = cS.size();

**if**(jumlah==0) {

System.***out***.println("Tidak ada komponen");

}**else** {

System.***out***.println("Jumlah Sisa Komponen = "+jumlah);

**for** (**int** i = 0; i <jumlah; i++) {

**if**(cS.get(i).getCheck()==1) {

System.***out***.println("Nama : "+cS.get(i).getCPU() +" (Tipe : CPU)");

}**else** **if**(cS.get(i).getCheck()==2) {

System.***out***.println("Nama : "+cS.get(i).getAccelarator() +" (Tipe : Accelerator)");

}**else** **if**(cS.get(i).getCheck()==3) {

System.***out***.println("Nama : "+cS.get(i).getRAM() +" (Tipe : RAM)");

}**else** **if**(cS.get(i).getCheck()==4) {

System.***out***.println("Nama : "+cS.get(i).getCache() +" (Tipe : Cache)");

}**else** **if**(cS.get(i).getCheck()==5) {

System.***out***.println("Nama : "+cS.get(i).getAdditionalCard() +" (Tipe : Additional Card)");

}

}

System.***out***.println();

}

}

}

**Memory.java**

**import** java.util.ArrayList;

**public** **abstract** **class** Memory **extends** Computer{

**protected** **int** maxSpeed = 50;

**public** **int** getMaxSpeed() {

**return** maxSpeed;

}

**public** **void** setMaxSpeed(**int** maxSpeed) {

**this**.maxSpeed = maxSpeed;

}

@Override

**public** **boolean** canFitInSlot(ComputerSlot slot) {

**return** **true**;

}

**public** **abstract** **void** canRead();

**public** **abstract** **void** canWrite();

**public** **abstract** **void** delMemory(String name,ArrayList<ComputerSlot> cS);

**public** **abstract** **void** printComputerComponent(ArrayList<ComputerSlot> cS);

}

**RAM.java**

**import** java.util.ArrayList;

**public** **class** RAM **extends** Memory{

**void** addRAM(String RAM, ArrayList<ComputerSlot> cS) {

**if**(canFitInSlot(**null**) ==**true**) {

**int** hitung = 0;

System.***out***.println("Berhasil Insert RAM dengan Nama "+RAM);

**for** (**int** i = 0; i < cS.size(); i++) {

**if**(cS.get(i).getCheck()==3) {

hitung++;

}

}

**if**(hitung ==0) {

cS.add(**new** ComputerSlot("", "", RAM, "", "", 0, getMaxSpeed()+10, 3));

System.***out***.println("Maksimal Speed Menjadi "+cS.get(cS.size()-1).getComputerSpeed());

System.***out***.println();

}**else** {

cS.add(**new** ComputerSlot("", "", RAM, "", "", 0, cS.get(cS.size()-1).getComputerSpeed()+10, 3));

System.***out***.println("Maksimal Speed Menjadi "+cS.get(cS.size()-1).getComputerSpeed());

System.***out***.println();

}

}

}

@Override

**public** **void** canRead() {

System.***out***.println("RAM Berhasil Membaca");

}

@Override

**public** **void** canWrite() {

System.***out***.println("RAM Berhasil Menulis");

}

@Override

**public** **void** delMemory(String name, ArrayList<ComputerSlot> cS) {

**int** count = cS.size();

**int** indeks = -1;

**for** (**int** i = 0; i < count; i++) {

**if**(cS.get(i).getRAM().equals(name)) {

indeks = i;

**break**;

}

}

**if**(indeks >= 0) {

cS.remove(indeks);

System.***out***.println("Berhasil Hapus Komponen dengan nama "+name);

}

}

@Override

**public** **void** printComputerComponent(ArrayList<ComputerSlot> cS) {

**int** jumlah = cS.size();

**if**(jumlah==0) {

System.***out***.println("Tidak ada komponen");

}**else** {

System.***out***.println("Jumlah Sisa Komponen = "+jumlah);

**for** (**int** i = 0; i <jumlah; i++) {

**if**(cS.get(i).getCheck()==1) {

System.***out***.println("Nama : "+cS.get(i).getCPU() +" (Tipe : CPU)");

}**else** **if**(cS.get(i).getCheck()==2) {

System.***out***.println("Nama : "+cS.get(i).getAccelarator() +" (Tipe : Accelerator)");

}**else** **if**(cS.get(i).getCheck()==3) {

System.***out***.println("Nama : "+cS.get(i).getRAM() +" (Tipe : RAM)");

}**else** **if**(cS.get(i).getCheck()==4) {

System.***out***.println("Nama : "+cS.get(i).getCache() +" (Tipe : Cache)");

}**else** **if**(cS.get(i).getCheck()==5) {

System.***out***.println("Nama : "+cS.get(i).getAdditionalCard() +" (Tipe : Additional Card)");

}

}

System.***out***.println();

}

}

}

**Cache.java**

**import** java.util.ArrayList;

**public** **class** Cache **extends** Memory{

**void** addCache(String Cache, ArrayList<ComputerSlot> cS) {

**if**(canFitInSlot(**null**) ==**true**) {

System.***out***.println("Berhasil Insert Cache dengan Nama "+Cache);

**if**(cS.size()==0) {

cS.add(**new** ComputerSlot("", "", "", Cache, "", 0, getMaxSpeed() + 10, 4));

System.***out***.println("Maksimal Speed Menjadi "+cS.get(cS.size()-1).getComputerSpeed());

System.***out***.println();

}**else** {

cS.add(**new** ComputerSlot("", "", "", Cache, "", 0, cS.get(cS.size()-1).getComputerSpeed()+10, 4));

System.***out***.println("Maksimal Speed Menjadi "+cS.get(cS.size()-1).getComputerSpeed());

System.***out***.println();

}

}

}

@Override

**public** **void** canRead() {

System.***out***.println("Cache Berhasil Membaca");

}

@Override

**public** **void** canWrite() {

System.***out***.println("Cache Berhasil Menulis");

}

@Override

**public** **void** delMemory(String name, ArrayList<ComputerSlot> cS) {

**int** count = cS.size();

**int** indeks = -1;

**for** (**int** i = 0; i < count; i++) {

**if**(cS.get(i).getCache().equals(name)) {

indeks = i;

**break**;

}

}

**if**(indeks >= 0) {

cS.remove(indeks);

System.***out***.println("Berhasil Hapus Komponen dengan nama "+name);

}

}

@Override

**public** **void** printComputerComponent(ArrayList<ComputerSlot> cS) {

**int** jumlah = cS.size();

**if**(jumlah==0) {

System.***out***.println("Tidak ada komponen");

}**else** {

System.***out***.println("Jumlah Sisa Komponen = "+jumlah);

**for** (**int** i = 0; i <jumlah; i++) {

**if**(cS.get(i).getCheck()==1) {

System.***out***.println("Nama : "+cS.get(i).getCPU() +" (Tipe : CPU)");

}**else** **if**(cS.get(i).getCheck()==2) {

System.***out***.println("Nama : "+cS.get(i).getAccelarator() +" (Tipe : Accelerator)");

}**else** **if**(cS.get(i).getCheck()==3) {

System.***out***.println("Nama : "+cS.get(i).getRAM() +" (Tipe : RAM)");

}**else** **if**(cS.get(i).getCheck()==4) {

System.***out***.println("Nama : "+cS.get(i).getCache() +" (Tipe : Cache)");

}**else** **if**(cS.get(i).getCheck()==5) {

System.***out***.println("Nama : "+cS.get(i).getAdditionalCard() +" (Tipe : Additional Card)");

}

}

System.***out***.println();

}

}

}

**AdditionalCard.java**

**import** java.util.ArrayList;

**public** **abstract** **class** AdditionalCard **extends** Computer{

@Override

**public** **boolean** canFitInSlot(ComputerSlot slot) {

**return** **true**;

}

**public** **abstract** **void** addCard(String CPUName, ArrayList<ComputerSlot> cS,**int** papanEkstensi);

**public** **abstract** **void** canRead();

**public** **abstract** **void** canWrite();

**public** **abstract** **void** cardBehaviour();

**public** **abstract** **void** delAdditionalCard(String name,ArrayList<ComputerSlot> cS);

**public** **abstract** **void** printComputerComponent(ArrayList<ComputerSlot> cS);

}

**FlacAudio.java**

**import** java.util.ArrayList;

**public** **class** FlacAudio **extends** AdditionalCard{

@Override

**public** **void** canRead() {

System.***out***.println("Flac Audio Berhasil Membaca");

}

@Override

**public** **void** canWrite() {

System.***out***.println("Flac Audio Berhasil Menulis");

}

@Override

**public** **void** cardBehaviour() {

System.***out***.println("Memainkan Format Audio Flac");

}

@Override

**public** **void** addCard(String FlacAudio, ArrayList<ComputerSlot> cS, **int** papanEkstensi) {

**if**(canFitInSlot(**null**) ==**true**) {

**int** hitung = 1;

**for** (**int** i = 0; i < cS.size(); i++) {

**if**(cS.get(i).getCheck()==5) {

hitung++;

}

}

**if**(hitung <=3 && papanEkstensi==1) {

System.***out***.println("Berhasil Insert Flac Audio dengan Nama "+FlacAudio);

**if**(cS.size()==0) {

cS.add(**new** ComputerSlot("", "", "", "", FlacAudio,0 , 0, 5));

System.***out***.println();

}**else** {

cS.add(**new** ComputerSlot("", "", "", "", FlacAudio,0 , 0, 5));

System.***out***.println();

}

}**else** **if**(papanEkstensi==0) {

System.***out***.println("Tidak ada Papan Ekstensi, Tidak Bisa Insert Card!!!");

}

**else** {

System.***out***.println("Slot untuk Kartu Tambahan Sudah Penuh...");

}

}

}

@Override

**public** **void** delAdditionalCard(String name, ArrayList<ComputerSlot> cS) {

**int** count = cS.size();

**int** indeks = -1;

**for** (**int** i = 0; i < count; i++) {

**if**(cS.get(i).getAdditionalCard().equals(name)) {

indeks = i;

**break**;

}

}

**if**(indeks >= 0) {

cS.remove(indeks);

System.***out***.println("Berhasil Hapus Komponen dengan nama "+name);

}

}

@Override

**public** **void** printComputerComponent(ArrayList<ComputerSlot> cS) {

**int** jumlah = cS.size();

**if**(jumlah==0) {

System.***out***.println("Tidak ada komponen");

}**else** {

System.***out***.println("Jumlah Sisa Komponen = "+jumlah);

**for** (**int** i = 0; i <jumlah; i++) {

**if**(cS.get(i).getCheck()==1) {

System.***out***.println("Nama : "+cS.get(i).getCPU() +" (Tipe : CPU)");

}**else** **if**(cS.get(i).getCheck()==2) {

System.***out***.println("Nama : "+cS.get(i).getAccelarator() +" (Tipe : Accelerator)");

}**else** **if**(cS.get(i).getCheck()==3) {

System.***out***.println("Nama : "+cS.get(i).getRAM() +" (Tipe : RAM)");

}**else** **if**(cS.get(i).getCheck()==4) {

System.***out***.println("Nama : "+cS.get(i).getCache() +" (Tipe : Cache)");

}**else** **if**(cS.get(i).getCheck()==5) {

System.***out***.println("Nama : "+cS.get(i).getAdditionalCard() +" (Tipe : Additional Card)");

}

}

System.***out***.println();

}

}

}

**OggAudio.java**

**import** java.util.ArrayList;

**public** **class** OggAudio **extends** AdditionalCard{

@Override

**public** **void** canRead() {

System.***out***.println("Ogg Audio Berhasil Membaca");

}

@Override

**public** **void** canWrite() {

System.***out***.println("Ogg Audio Berhasil Menulis");

}

@Override

**public** **void** cardBehaviour() {

System.***out***.println("Memainkan Format Audio Ogg");

}

@Override

**public** **void** addCard(String OggAudio, ArrayList<ComputerSlot> cS, **int** papanEkstensi) {

**if**(canFitInSlot(**null**) ==**true**) {

**int** hitung = 1;

**for** (**int** i = 0; i < cS.size(); i++) {

**if**(cS.get(i).getCheck()==5) {

hitung++;

}

}

**if**(hitung <=3 && papanEkstensi==1) {

System.***out***.println("Berhasil Insert Ogg Audio dengan Nama "+OggAudio);

**if**(cS.size()==0) {

cS.add(**new** ComputerSlot("", "", "", "", OggAudio,0 , 0, 5));

System.***out***.println();

}**else** {

cS.add(**new** ComputerSlot("", "", "", "", OggAudio,0 , 0, 5));

System.***out***.println();

}

}**else** **if**(papanEkstensi==0) {

System.***out***.println("Tidak ada Papan Ekstensi, Tidak Bisa Insert Card!!!");

}

**else** {

System.***out***.println("Slot untuk Kartu Tambahan Sudah Penuh...");

}

}

}

@Override

**public** **void** delAdditionalCard(String name, ArrayList<ComputerSlot> cS) {

**int** count = cS.size();

**int** indeks = -1;

**for** (**int** i = 0; i < count; i++) {

**if**(cS.get(i).getAdditionalCard().equals(name)) {

indeks = i;

**break**;

}

}

**if**(indeks >= 0) {

cS.remove(indeks);

System.***out***.println("Berhasil Hapus Komponen dengan nama "+name);

}

}

@Override

**public** **void** printComputerComponent(ArrayList<ComputerSlot> cS) {

**int** jumlah = cS.size();

**if**(jumlah==0) {

System.***out***.println("Tidak ada komponen");

}**else** {

System.***out***.println("Jumlah Sisa Komponen = "+jumlah);

**for** (**int** i = 0; i <jumlah; i++) {

**if**(cS.get(i).getCheck()==1) {

System.***out***.println("Nama : "+cS.get(i).getCPU() +" (Tipe : CPU)");

}**else** **if**(cS.get(i).getCheck()==2) {

System.***out***.println("Nama : "+cS.get(i).getAccelarator() +" (Tipe : Accelerator)");

}**else** **if**(cS.get(i).getCheck()==3) {

System.***out***.println("Nama : "+cS.get(i).getRAM() +" (Tipe : RAM)");

}**else** **if**(cS.get(i).getCheck()==4) {

System.***out***.println("Nama : "+cS.get(i).getCache() +" (Tipe : Cache)");

}**else** **if**(cS.get(i).getCheck()==5) {

System.***out***.println("Nama : "+cS.get(i).getAdditionalCard() +" (Tipe : Additional Card)");

}

}

System.***out***.println();

}

}

}

**mp3Audio.java**

**import** java.util.ArrayList;

**public** **class** mp3Audio **extends** AdditionalCard{

@Override

**public** **void** canRead() {

System.***out***.println("MP3 Audio Berhasil Membaca");

}

@Override

**public** **void** canWrite() {

System.***out***.println("MP3 Audio Berhasil Menulis");

}

@Override

**public** **void** cardBehaviour() {

System.***out***.println("Memainkan Format Audio Mp3");

}

@Override

**public** **void** addCard(String mp3Audio, ArrayList<ComputerSlot> cS, **int** papanEkstensi) {

**if**(canFitInSlot(**null**) ==**true**) {

**int** hitung = 1;

**for** (**int** i = 0; i < cS.size(); i++) {

**if**(cS.get(i).getCheck()==5) {

hitung++;

}

}

**if**(hitung <=3 && papanEkstensi==1) {

System.***out***.println("Berhasil Insert Mp3 Audio dengan Nama "+mp3Audio);

**if**(cS.size()==0) {

cS.add(**new** ComputerSlot("", "", "", "", mp3Audio,0 , 0, 5));

System.***out***.println();

}**else** {

cS.add(**new** ComputerSlot("", "", "", "", mp3Audio,0 , 0, 5));

System.***out***.println();

}

}**else** **if**(papanEkstensi==0) {

System.***out***.println("Tidak ada Papan Ekstensi, Tidak Bisa Insert Card!!!");

}

**else** {

System.***out***.println("Slot untuk Kartu Tambahan Sudah Penuh...");

}

}

}

@Override

**public** **void** delAdditionalCard(String name, ArrayList<ComputerSlot> cS) {

**int** count = cS.size();

**int** indeks = -1;

**for** (**int** i = 0; i < count; i++) {

**if**(cS.get(i).getAdditionalCard().equals(name)) {

indeks = i;

**break**;

}

}

**if**(indeks >= 0) {

cS.remove(indeks);

System.***out***.println("Berhasil Hapus Komponen dengan nama "+name);

}

}

@Override

**public** **void** printComputerComponent(ArrayList<ComputerSlot> cS) {

**int** jumlah = cS.size();

**if**(jumlah==0) {

System.***out***.println("Tidak ada komponen");

}**else** {

System.***out***.println("Jumlah Sisa Komponen = "+jumlah);

**for** (**int** i = 0; i <jumlah; i++) {

**if**(cS.get(i).getCheck()==1) {

System.***out***.println("Nama : "+cS.get(i).getCPU() +" (Tipe : CPU)");

}**else** **if**(cS.get(i).getCheck()==2) {

System.***out***.println("Nama : "+cS.get(i).getAccelarator() +" (Tipe : Accelerator)");

}**else** **if**(cS.get(i).getCheck()==3) {

System.***out***.println("Nama : "+cS.get(i).getRAM() +" (Tipe : RAM)");

}**else** **if**(cS.get(i).getCheck()==4) {

System.***out***.println("Nama : "+cS.get(i).getCache() +" (Tipe : Cache)");

}**else** **if**(cS.get(i).getCheck()==5) {

System.***out***.println("Nama : "+cS.get(i).getAdditionalCard() +" (Tipe : Additional Card)");

}

}

System.***out***.println();

}

}

}

**Main.java**

**import** java.util.ArrayList;

**public** **class** Main {

ArrayList<ComputerSlot> CS = **new** ArrayList<>();

**public** Main() {

//INSERT CPU

CPU cpu = **new** CPU();

cpu.addCPU("Fiona",CS);

cpu.addCPU("Varencia",CS);

cpu.addCPU("Tendio",CS);

cpu.delProcessor("Tendio",CS);

cpu.printComputerComponent(CS);

cpu.canRead();

cpu.canWrite();

System.***out***.println();

//INSERT ACCELERATOR

Accelerator accelerator = **new** Accelerator();

accelerator.addAccelerator("Fio",CS);

accelerator.addAccelerator("Varen",CS);

accelerator.delProcessor("Varen",CS);

accelerator.printComputerComponent(CS);

accelerator.canRead();

accelerator.canWrite();

System.***out***.println();

//INSERT RAM

RAM ram = **new** RAM();

ram.addRAM("Dio", CS);

ram.addRAM("Ten", CS);

ram.delMemory("Dio", CS);

ram.printComputerComponent(CS);

ram.canRead();

ram.canWrite();

//INSERT CACHE

Cache cache = **new** Cache();

cache.addCache("Cia", CS);

cache.addCache("Na", CS);

cache.addCache("Endi", CS);

cache.delMemory("Na", CS);

cache.printComputerComponent(CS);

//INSERT ADDITIONAL CARD FLAC

FlacAudio flacaudio = **new** FlacAudio();

flacaudio.addCard("Flac", CS, 1);

flacaudio.canRead();

flacaudio.canWrite();

flacaudio.printComputerComponent(CS);

//INSERT ADDITIONAL CARD OGG

OggAudio oggaudio = **new** OggAudio();

oggaudio.addCard("Ogg", CS, 1);

oggaudio.canRead();

oggaudio.canWrite();

oggaudio.printComputerComponent(CS);

//INSERT ADDITIONAL CARD MP3

mp3Audio mp3audio = **new** mp3Audio();

mp3audio.addCard("Mp3", CS, 1);

mp3audio.canRead();

mp3audio.canWrite();

mp3audio.printComputerComponent(CS);

//INSERT MORE THAN 3 ADDTIONAL CARD

flacaudio.addCard("Flac1", CS, 1);

//INSERT BUT NO EXTENSION CARD

oggaudio.addCard("Ogg1", CS, 0);

}

**public** **static** **void** main(String argvs[]) {

**new** Main();

}

}

1. **[LO 2, 10 poin]** Berdasarkan soal no.2, gambarkan kelas diagram dari sistem yang ada. Kelas diagram yang dibuat sudah menggabungkan design pattern yang diterapkan pada soal no. 2!

**Jawaban:**

Diagram

Description automatically generated