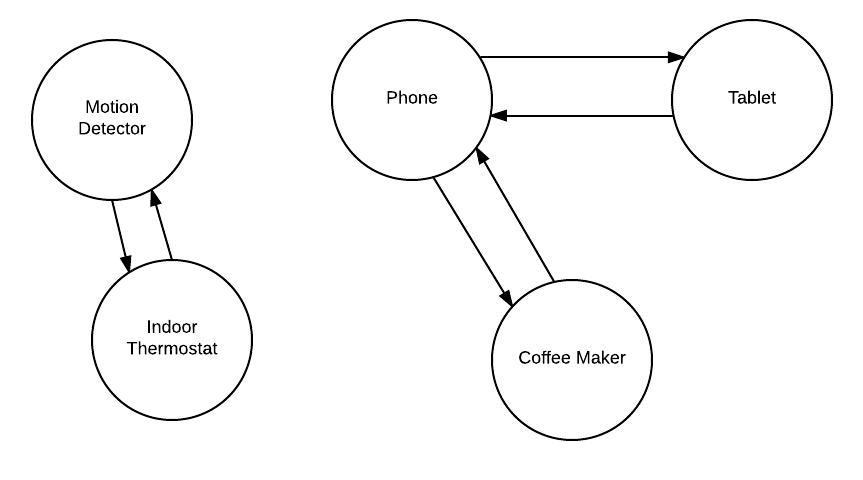
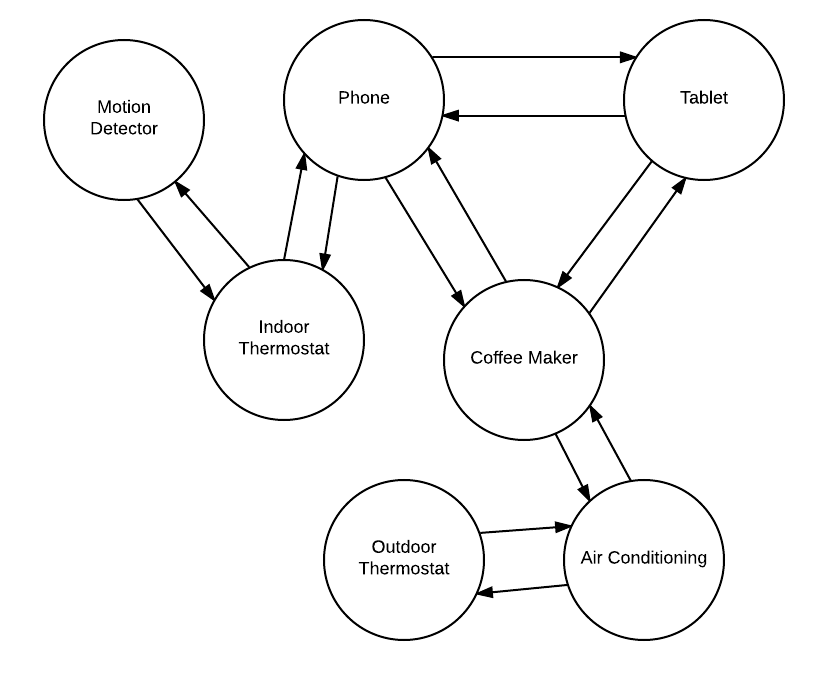
* **Mediator** – ofera un mediu centralizat de comunicare intre diferite obiecte
* Intr-un sistem, adesea obiectele comunica intre ele. Totusi, daca in fiecare obiect am avea referinte la toate celelalte si am defini metode pentru a comunica cu fiecare, codul ar fi urias si greu de inteles
* Mediator e cel care asigura comunicarea intre obiecte, deci fiecare obiect va comunica cu altele nu direct prin referinte la ele, ci prin Mediator
* **Collegues** – obiecte care comunica intre ele prin mediator

**Exemplu**

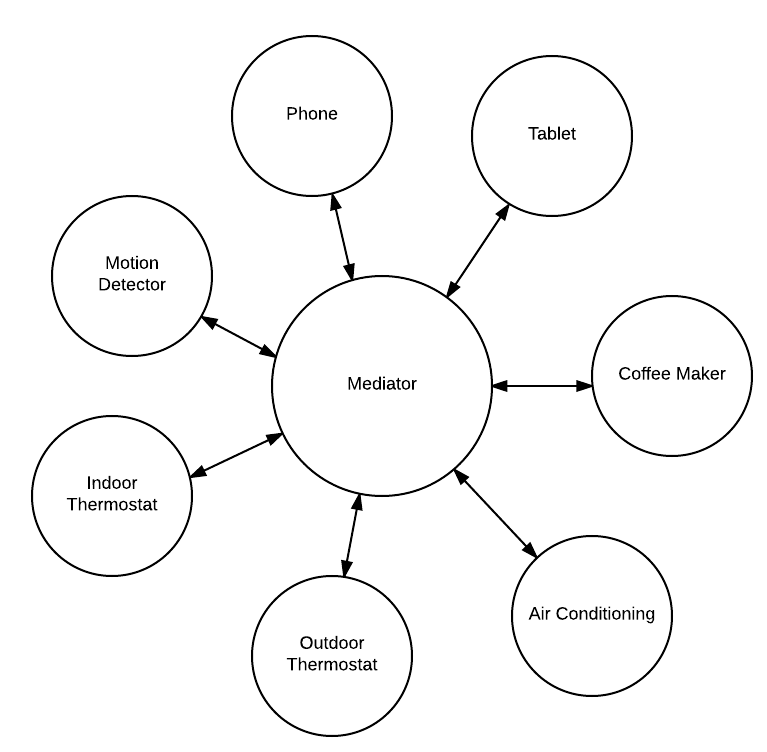
* Sa zicem ca facem o casa a viitorului.
* Casa isi va modifica temperatura odata ce plecam, va face cafeaua odata ce alarma telefonului suna, si va incarca cel mai recent email pe tableta daca suntem acasa si e Sambata dimineat



Deocamdata totul e ok si usor, obiectele pot comunica direct unele cu altele. Dar, daca mai adaugam functii, totul poate deveni prea complicat:

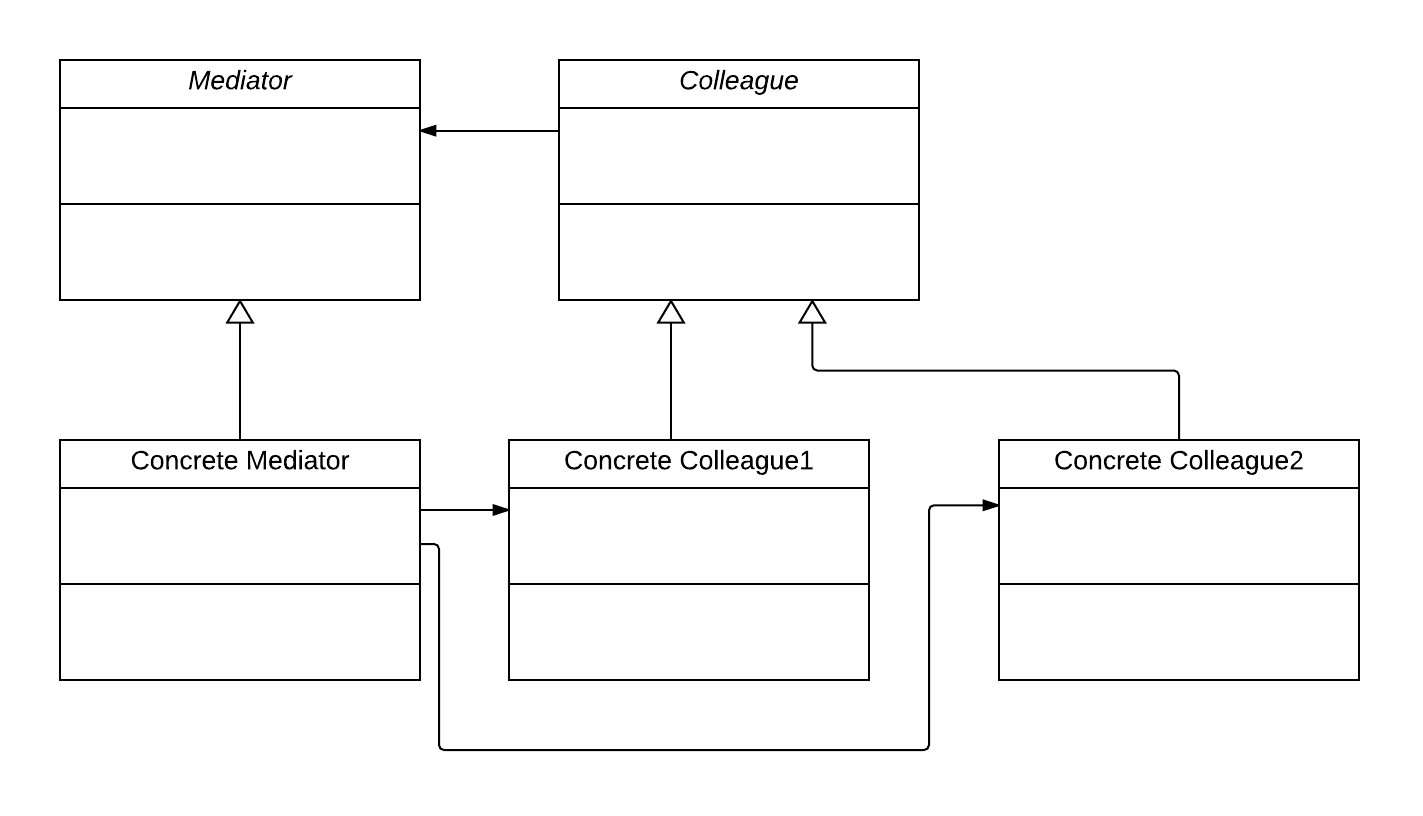


Mediator acum e solutia.



Acum, fiecare obiect va avea doar un field si anume la mediator. Deja mediator va avea referinte la toate obiectele si va face comunicarea intre ele.

* Fiecare obiect, ca sa comunice cu altul, va informa mediatorul, va spune cu care vrea sa comunice si mediator va face totul.



**Exemplu 2**

Fie o aplicatie in care userii vorbesc pe chat intre ei, pe un grup. Un user poate trimite mesaje pe grup, deci la toti userii

Aici, nu avem prorpiu zis mai multe clase ce colaboreaza intre ele, dar mai multe obiecte de acelasi tip. Si asa ceva e posibil. Oricum, s-ar primi ca fiecare User ar trebui sa aiba lista cu toti userii existenti, si asta deja e foarte rau.

public interface ChatMediator {

public void sendMessage(String msg, User user);

void addUser(User user);

}

public abstract class User {

protected ChatMediator mediator;

protected String name;

public User(ChatMediator med, String name){

this.mediator=med;

this.name=name;

}

public abstract void send(String msg);

public abstract void receive(String msg);

}

Vedem ca fiecare User are o referinta catre Mediator

Asta e mediator

public class ChatMediatorImpl implements ChatMediator {

private List<User> users;

public ChatMediatorImpl(){

this.users=new ArrayList<>();

}

@Override

public void addUser(User user){

this.users.add(user);

}

@Override

public void sendMessage(String msg, User user) {

for(User u : this.users){

//message should not be received by the user sending it

if(u != user){

u.receive(msg);

}

}

}

}

public class UserImpl extends User {

public UserImpl(ChatMediator med, String name) {

super(med, name);

}

@Override

public void send(String msg){

System.out.println(this.name+": Sending Message="+msg);

mediator.sendMessage(msg, this);

}

@Override

public void receive(String msg) {

System.out.println(this.name+": Received Message:"+msg);

}

}

public class ChatClient {

public static void main(String[] args) {

ChatMediator mediator = new ChatMediatorImpl();

User user1 = new UserImpl(mediator, "Pankaj");

User user2 = new UserImpl(mediator, "Lisa");

User user3 = new UserImpl(mediator, "Saurabh");

User user4 = new UserImpl(mediator, "David");

mediator.addUser(user1);

mediator.addUser(user2);

mediator.addUser(user3);

mediator.addUser(user4);

user1.send("Hi All");

}

}

Pankaj: Sending Message=Hi All

Lisa: Received Message:Hi All

Saurabh: Received Message:Hi All

David: Received Message:Hi All

Observam ca obiectele trimit propria referinta catre mediator. Asta e necesar.