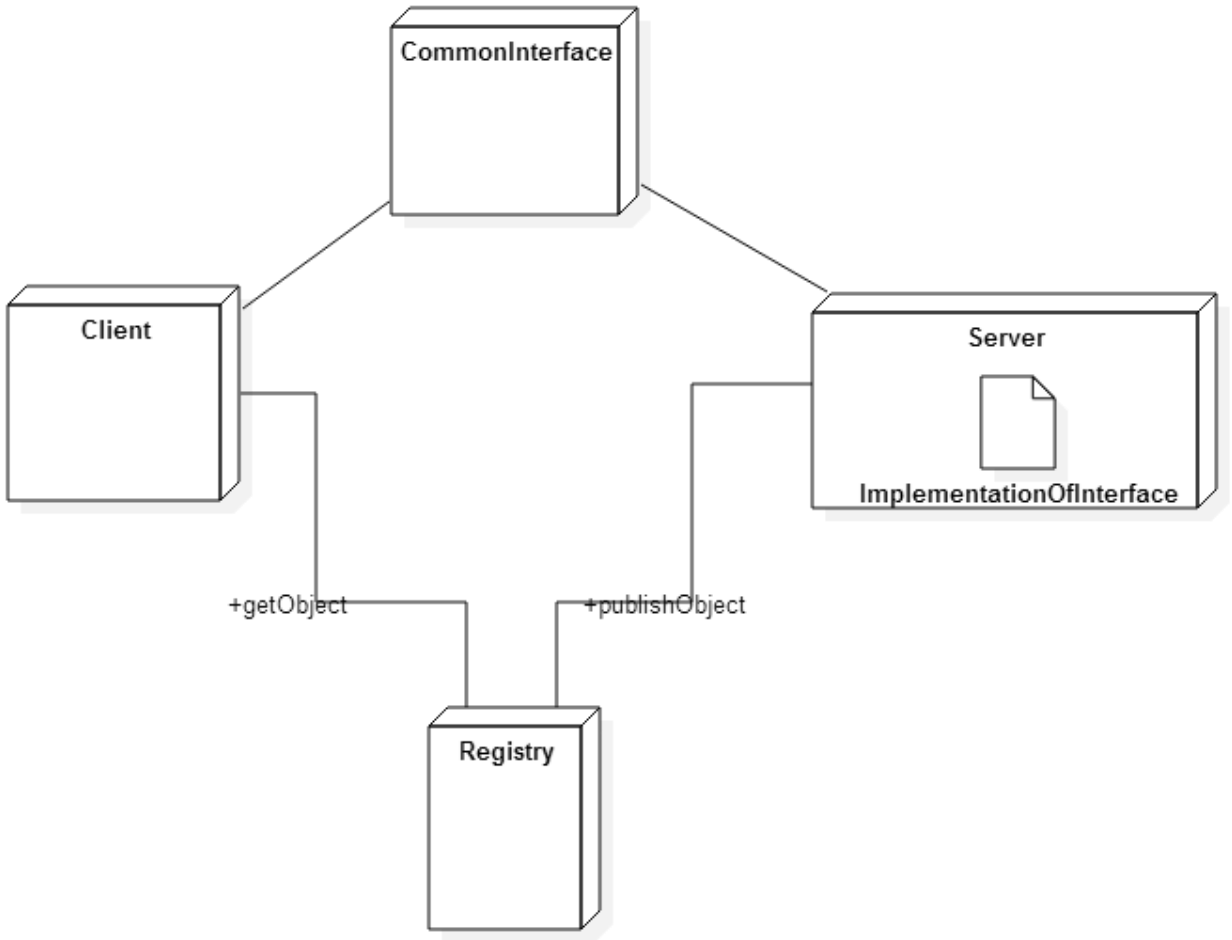
Assignment 2.2

Tolas Ramona Ioana

Grupa 30243

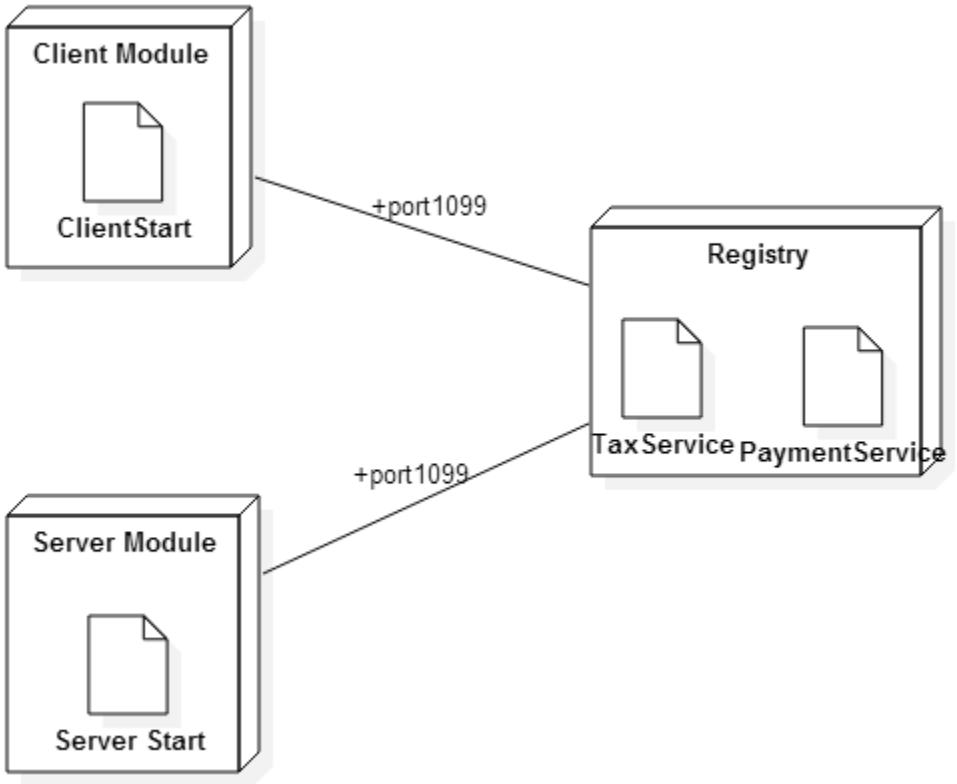
1. Arhitectura conceptuala

Aplicatia este de tipul Client-Server avand astfel doua componente principale care comunica. Remote method invocation este conceptul folosit. Acest lucru presupune ca o entitate a aplicatiei (Clientul) foloseste o metoda, un obiect care nu este pe aceeasi masina cu el. Necesitatea implementarii acestei arhitecturi vine de la avantajele pe care acesta le ofera. Printre acestea se enumara posibilitatea de a pune pe masini diferite clientul si serverul. Acest lucru este o buna practica atunci cand algoritmul care ruleaza pe server necesita resurse foarte multe, resurse pe care clientul nu si le permite sa le aiba. In acest mod, clientul mai slab in resurse trimite o cerere la server, fara macar sa stie ca algoritmul nu se executa de catre el. Intre Client si Server se gaseste Registry. Acesta reprezinta o parte a aplicatiei in care serverul isi publica obiectele si pe care clientul le cere spre folosinta. Pentru a sti Clientul ce fel de obiecte sunt publicate in registry si ce pot acestea sa faca exista o interfata comuna pe care atat Clientul cat si Serverul le cunosc. Pentru a exemplifica aceasta arhitectura s-a implementat calcul unei taxe pentru o anumita masina. Clientul da informatiile necesare pentru a calcula taxa si asteapta de la server raspunsul fara sa execute nici un calcul el. Ceea ce se implementeaza de catre server este o operatie banala si nu necesita multa putere de calcul, dar exemplul este doar pentru a intelege modalitatea de gstiune a resurselor in cadrul uneia arhitecturi ce foloseste Remote Method Invocation.



2. Diagrama UML de deployment

Dupa cum s-a mentionat in capitolul anterior, RMI este folosita in principal pentru a separa Clientul de Server atunci cand Clientul nu are destule resurse pentru a face algoritmul. Pentru a simula aceasta separare (in mod normal cele doua enitati sunt pe masini diferite) proiectul foloseste mai multe module. Astfel un modul este reprezentat de Client, un altul de Server . Comunicarea intre cele se face prin intermediul Registry si nu in mod direct. Registry are nevoie de un port pe care sa comunice cu cele doua entitati. Portul trebuie sa fie stiut atat de server cat si de Client. In cazul java RAMI , portul implicit este 1099, dar se poate folosi orice port.



3. Manual de utilizare

Pentru a se folosi aplicatia se deascarca arhiva si se dezarhiveaza. SE urmeaza apoi urmatorii pasi :

- Se deschide un mediu de dezvoltare (spre exemplu Eclipse)
- Se importa proiectul ca si proiect Maven (in acest mod se vor importa toate modulele)
- Se ruleaza Serverul (prin rularea metodei main din clasa Server Start). Daca totul este in regula in consola va aparea mesajul *: Server ready.* La pornirea Serverului se porneste automat si Registry deci nu mai este nevoie de nici un demers pentru a pune in functiune si acesta componenta a aplicatiei.
- Se porneste Clientul prin rularea clasei Client Start. Va aparea un form in care sunt necesare sa se introduca datele pentru identificarea masinii. Se completeaza datele cerute si se apasa butonul *GET INFO.* In zona de text va aparea raspunsul cu taxa si pretul de vanzare daca datele introduse au fost corecte sau va aparea un mesaj de eroare in cazul in care datele introduse nu au fost corecte.