Laboratorium 6

Wstęp

Technologia RMI jest technologią pozwalającą na zdalne wywoływanie metod, tzn. na realizację architektury klient serwer. Pozwala to na proste stworzenie rozproszonego systemu komunikującego się między sobą.

Podstawy:

W celu wykorzystania technologii RMI niezbędne jest stworzenie interfejsu rozszerzającego interfejs Remote. Uwaga!!! Każda metoda tego interfejsu musi zwracać wyjątek RemoteException

Przykład:

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface JakisInterfejs extends Remote{
    public void wyslij(String s) throws RemoteException;
}
```

Drugim elementem jest stworzenie implementacji tego interfejsu:

```
import java.rmi.RemoteException;

public class ImplementacjaInterfejsu implements JakisInterfejs{

    @Override
    public void wyslij(String s) throws RemoteException {
        System.out.println("Odebrałem " + s);
    }

}
```

Pozostaje jeszcze stworzenie aplikacji klienta i serwera.

Serwer:

```java
import java.rmi.RMISecurityManager;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;

public class Serwer {
    public static void main(String[] args) {
        if (System.getSecurityManager() == null) {
            System.setSecurityManager(new RMISecurityManager());
        }
        try {
            String name = "Moj Zdalny Przyklad";
            JakisInterfejs implementacjaSerwera = new ImplementacjaInterfejsu();
            JakisInterfejs stub =
                (JakisInterfejs) UnicastRemoteObject.exportObject(implementacjaSerwera,0);
            Registry registry = LocateRegistry.createRegistry(1099);
            //Registry registry = LocateRegistry.getRegistry();
            registry.rebind(name, stub);
            System.out.println("Uruchomiono i podpięto zdalnyKalkulator");
        catch (Exception e) {
            System.err.println("Wyjatek podczas włączania:");
            e.printStackTrace();
        }
    }
}
```

Tworzenie menadżera bezpieczeństwa

Tworzenie instancji zdalnego obiektu (będzie ona odpowiadała za realizację zadań)

Przygotowanie obiektu do zdalnych obliczeń

Pobranie lub stworzenie rejestratora nazw

Skojarzenie rejestratora nazw z określoną przez nas nazwą

Klient:

```java
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Klient {
    public static void main(String[] args) {
        if (System.getSecurityManager() == null) {
            System.setSecurityManager(new SecurityManager());
        }
        {
            String name = "Moj Zdalny Przyklad";
            Registry registry = LocateRegistry.getRegistry();
            //Registry registry = LocateRegistry.getRegistry(host, port);
            JakisInterfejs zdalnyObiekt = (JakisInterfejs)registry.lookup(name);
            System.out.println("Przygotowano program do działania");
            zdalnyObiekt.wyslij("Hello serwer");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Tworzenie menadżera bezpieczeństwa

Pobranie rejestratora nazw zdalnych obiektów. Jeśli rejestr na innym hoście to podać adres do hosta

Pobranie z rejestratora „zdalnego wskaźnika” Do obiektu działającego na serwerze

Do poprawnej pracy przed uruchomieniem konieczne jest jeszcze przygotowanie warunków bezpieczeństwa dla aplikacji serwera i klienta. W tym celu konieczne są pliki: *client.policy* i *server.policy*. Określają one uprawnienia klienta i serwera. W najprostszej formie mają one postać:

```
grant {
   permission java.security.AllPermission;
};
```

Co oznacza że aplikacji serwera i klienta przyznajemy pełnię praw. Uwaga!!! Takie uprawnienia mogą być niebezpieczne!!!

Aby określone uprawnienia zostały wczytane uruchamiając wirtualną maszynę javy dla klienta i serwera należy podać dodatkowy parametr *-Djava.security.policy=client.policy* określający nazwę pliku z uprawnieniami. Tzn: *java -Djava.security.policy=server.policy Serwer,* a następnie : *java -Djava.security.policy=client.policy Klient*

Zadanie 1

Korzystając z technologii RMI stwórz zdalny kalkulator który realizuje metody dodawania, odejmowania, mnożenia i dzielenia.

Serwer powinien pod jedną nazwą w rejestrze implementować wszystkie powyższe metody. Metody jako argumenty powinny przyjmować liczby typu double oraz zwracać wynik operacji również jako double

Na jednostce kliencie, po dokonaniu połączenia, wczytaj dwie liczby z klawiatury i zapisz je do zmiennych *a* i *b* a następnie wyświetl na ekranie wyniki wszystkich możliwych operacji:

```
System.out.printf("Operacje na liczbach %f, i  %f \n",a,b);
System.out.printf("Dodawanie %f \n",kalkulator.dodaj(a, b));
System.out.printf("Mnozenie %f \n",kalkulator.mnoz(a, b));
System.out.printf("Dzielenie %f \n",kalkulator.dziel(a, b));
System.out.printf("Odejmowanie %f \n",kalkulator.odejmij(a, b));
```

Zadanie 2

Zastanów się jak zrobić uniwersalny system, który pozwala na wykonanie dowolnych (uniwersalnych) zadań po stronie serwera. Następnie zaimplementuj swoje rozwiązanie. W kolejnym kroku w oparciu o opracowane rozwiązanie każ serwerowi policzyć liczbę Pi metodą monteCarlo.

Metoda ta polega losowym generowaniu punktów należących do kwadratu o boku a, a następnie na wyznaczeniu liczby punktów które trafiają do koła wpisanego w ten kwadrat. Liczba Pi jest wówczas równa stosunkowi

Pi = n/m

Gdzie

n – liczba trafień wylosowanych liczb do koła

m – liczba wylosowanych liczb

Rozwiązanie zadania:

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ZdalneObliczenia extends Remote{
    void policz(Zadanie t) throws RemoteException;
    Object wynikObliczen() throws RemoteException;
}
```

I

```
public interface Zadanie {
    public void licz();
    public Object getWynik();
}
```