# ЛАБОРАТОРНАЯ РАБОТА №3

## УДАЛЕННЫЙ ВЫЗОВ МЕТОДОВ (RMI)

**Цель работы**: изучить основные понятия и навыки технологии RMI. Разработать клиент RMI и сервер RMI.

**Программное обеспечение:** IntelliJ IDEA, JDK 1.8+.

**Необходимая теоретическая подготовка:**

- объектно-ориентированное программирование;
- примитивные и ссылочные типы данных, классы-обертки;
- технология сериализация и десериализация;
- работа с сокетами на Java;
- технология RMI.

## ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Java RMI (Remote Method Invocation – удаленный вызов методов) представляет собой тип удаленного вызова процедур, независимый от сети, облегченный и полностью переносимый, так как написан на языке Java.

*Материалы лекции «Сериализация и RMI»*

### Пример реализации клиента RMI

```
public class Client {
   // Host or IP of Server
   private static final String HOST = "localhost";
   private static final int PORT = 1099;
   private static Registry registry;

   public static void main(String[] args) throws Exception {
       // Search the registry in the specific Host, Port.
       registry = LocateRegistry.getRegistry(HOST, PORT);
       // Lookup WeatherService in the Registry.
       WeatherService service = (WeatherService) registry
               .lookup(WeatherService.class.getSimpleName());
       Date today = new Date();
       // Get Chicago weather info:
       WeatherData chicagoWeather = service.getWeather(today,
               Constants.LOCATION_CHICAGO);
       System.out.println("Chicago weather today: "
               + chicagoWeather.getWeather());
       // Get Hanoi weather info:
       WeatherData hanoiWeather = service.getWeather(today,
               Constants.LOCATION_HANOI);
       System.out.println("Hanoi weather today: " +
hanoiWeather.getWeather());
   }
}
```

## Пример реализации сервера RMI

```
public class Server {
    private static final int PORT = 1099;
    private static Registry registry;
    public static void startRegistry() throws RemoteException {
        // Create server registry
        registry = LocateRegistry.createRegistry(PORT);
    }
    public static void registerObject(String name, Remote remoteObj)
            throws RemoteException, AlreadyBoundException {
        registry.bind(name, remoteObj);
        System.out.println("Registered: " + name + " -> "
                + remoteObj.getClass().getName() + "[" + remoteObj + "]");
    }

    public static void main(String[] args) throws Exception {
        System.out.println("Server starting...");
        startRegistry();
        registerObject(WeatherService.class.getSimpleName(), new
WeatherServiceImpl());
        // Server was the start, and was listening to the request from the
client.
        System.out.println("Server started!");
    }
}
```

## Порядок выполнения работы

1. изучить предлагаемый теоретический материал;
2. разработать сервер и клиент RMI;
3. запустить сервер (убедитесь, что на сервере запущен RMI реестр);
4. запустить клиента и передать на сервер в качестве параметра удаленного метода объект (инд. зад. л.р. №2);
5. при разработке сервера и клиента должны быть использованы соглашения об оформлении кода *java code convention*.

## Задания к лабораторной работе

Разработать клиент RMI и сервер RMI. Вызвать из клиента удаленный метод на сервере и передать в качестве параметра объект, класс которого был реализован в лабораторной работе №2.