Exercise – Remote Method Invocation (RMI)

- This exercise consists of two parts:
  1. You should create a simple distributed object application using one of the object-oriented programming languages that support RMI. This part helps you better understand how a typical RMI system is used by developers and reinforce your understanding of fundamentals of RMI.
  2. You need to design and implement a simple RMI middleware **without using the selected languages' RMI related packages**.
- Implementation of both parts should be done using Java, C#, or Python.

Exercise – Part 1

- Design and implement a calendar server that allows adding/removing events by clients using RMI. The calendar is the remote object that client codes invoke its methods. There can be more than one client invoking the remote methods of the calendar.
- Events that are added have a type (e.g. speech, tutorial, contest, sport match, etc.), date and time, length in minutes, name (unique), description, and location.

Exercise – Part 1

- Callender allows clients to rederive events based on their name, type, date, and location.
- Also, calendar has a remote method that returns all events for a given month and year.
- **Read** the following to better understand the Java RMI:

https://docs.oracle.com/javase/6/docs/technotes/guides/rmi/hello/hello-world.html

Exercise – Part 2

- In this part, no libraries that implement RPC/RMI should be used (e.g. Java RMI, Python PYRO, etc.)
- The implemented RMI should have the following features:
- Obviously should allow invocation of remote methods!
- To the developer, the remote method invocation should look like a local method call.
- You can not use classes in the Java RMI packages (only exception to this is the RemoteException!)

Exercise – continued

- Multiple clients should be able to connect to remote objects
- Should use inheritance to enforce method implementation and consistency at client and server sides.
- Follow good programming practices in designing your application.
- The binding of client and server objects should be done **dynamically**.

Exercise – continued

- Should support at least the following primitive types for arguments and return type: integer, boolean, char, String, and double.
- Support for objects as arguments and return types is optional.
- If you are making assumptions, clearly specify them in your report.

Exercise – continued

- After implementing the necessary classes/interfaces and designing the structure of your solution. create the calendar in part 1 using your solution.

Report

- Submit report by 23:55 on Tuesday 21st of Azar
- Create 2 short (maximum 5 min each) videos that shows how you did each part.
- In your report, briefly (2-3) describe how you did part 1.
- Your report should describe your solution for part 2 and your evaluation of it (its strength and weaknesses)
- Furthermore, describe your calendar implementation in part 2. Use class diagrams to describe structure of your solution.
- PDF file of the report, two zip files containing the source codes for each part, and 2 videos should be submitted before the deadline.