PROGRAMOWANIE ZAAWANSOWANE

Julia Kaczyńska 160040

Karolina Kozłowska 159824

Link do github: julia160040/Programowanie_zaawansowane (github.com)

Kod: Kasa Klient

```
import java.io.*;
import java.net.*;
import java.util.List;

public class Klient {
   public static void main(String[] args) {
      for (int i = 1; i <= 5; i++) {
         final int clientId = i;
         new Thread(() -> {
            requestObjects(clientId, "Kot");
            requestObjects(clientId, "Pies");
            requestObjects(clientId, "Ryba");
         }).start();
      }
   }

   private static void requestObjects(int clientId, String objType) {
      try (Socket socket = new Socket("127.0.0.1", 1899);
           ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());
           ObjectInputStream in = new ObjectInputStream(socket.getInputStream())) {

         out.writeInt(clientId);
         out.flush();

         String status = (String) in.readObject();
```

```
        if ("REFUSED".equals(status)) {
            System.out.println("Klient " + clientId + ": Odmowa połączenia");
            return;
        } else {
            System.out.println("Klient " + clientId + ": Połączenie zaakceptowane");
        }

        for (int i = 0; i < 3; i++) {
            out.writeObject("get_" + objType);
            out.flush();

            Object response = in.readObject();
            if (response instanceof String) {
                System.out.println("Klient " + clientId + ": " + response);
            } else {
                List<?> objects = (List<?>) response;
                System.out.println("Klient " + clientId + " Otrzymano " + objType + " Kolekcja: " +
objects);
            }
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

Kod: Klasa Pies

```
import java.io.Serializable;
public class Pies implements Serializable {
   private String rasa;

   public Pies(String rasa){
      this.rasa=rasa;
   }

   public String getRasa() {
      return rasa;
   }
  @Override
  public String toString(){
      return "Pies{" + "rasa: '" + rasa + '\'' + '}';
  }
}
```

```
import java.io.Serializable;
public class Kot implements Serializable{
   private String imie;

   public Kot(String imie){
      this.imie=imie;
   }
```

```
    public String getImie() {
        return imie;
    }
    @Override
    public String toString(){
        return "Kot{" + "imie: '" + imie + '\'' + '}';
    }
}
------------------------------
import java.io.Serializable;
public class Ryba implements Serializable {
    private String gatunek;

    public Ryba(String gatunek){
        this.gatunek=gatunek;
    }

    public String getGatunek() {
        return gatunek;
    }
    @Override
    public String toString(){
        return "Ryba{" + "gatunek: '" + gatunek + '\'' + '}';
    }
}
```

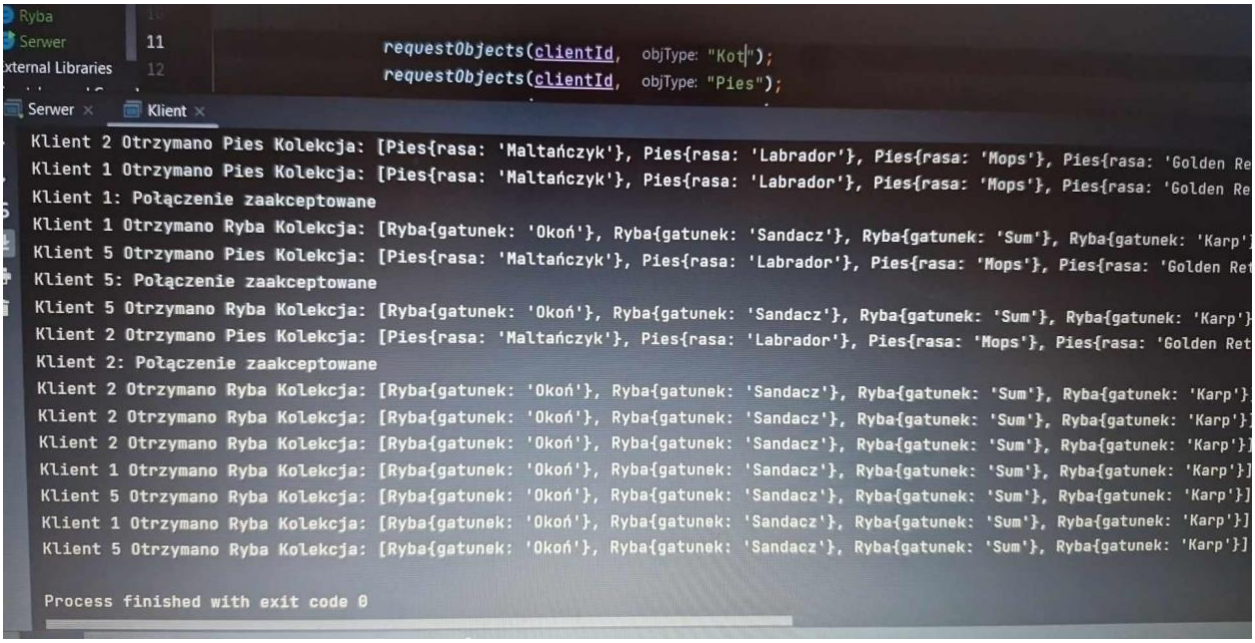
Kod: Klasa Serwer

```
import java.io.*;

import java.net.*;

import java.util.*;

import java.util.concurrent.*;


public class Serwer {

   private static final int MAX_CLIENTS = 5;

   private static final Map<String, List<Serializable>> objectsMap = new
ConcurrentHashMap<>();

   private static final Set<Integer> activeClients = Collections.newSetFromMap(new
ConcurrentHashMap<>());


   public static void main(String[] args) throws IOException {

      objectsMap.put("Pies", Arrays.asList(

            new Pies("Maltańczyk"),

            new Pies("Labrador"),

            new Pies("Mops"),

            new Pies("Golden Retriver")

      ));

      objectsMap.put("Kot", Arrays.asList(

            new Kot("Filemon"),

            new Kot("Bonifacy"),

            new Kot("Klakier"),

            new Kot("Garfild")

      ));
```

```
    objectsMap.put("Ryba", Arrays.asList(
        new Ryba("Okoń"),
        new Ryba("Sandacz"),
        new Ryba("Sum"),
        new Ryba("Karp")
    ));

    ServerSocket serverSocket = new ServerSocket(1899);
    System.out.println("Serwer uruchomiony na porcie 1899");

    while (true) {
        Socket clientSocket = serverSocket.accept();
        new Thread(new ClientHandler(clientSocket)).start();
    }
}

private static class ClientHandler implements Runnable {
    private final Socket clientSocket;
    private int clientId;

    public ClientHandler(Socket clientSocket) {
        this.clientSocket = clientSocket;
    }

    @Override
    public void run() {
```

```
        try (ObjectOutputStream out = new
ObjectOutputStream(clientSocket.getOutputStream());
             ObjectInputStream in = new ObjectInputStream(clientSocket.getInputStream())) {

            clientId = in.readInt();
            synchronized (activeClients) {
               if (activeClients.size() >= MAX_CLIENTS) {
                  out.writeObject("ODMOWA");
                  System.out.println("Klient " + clientId + " Odmowa: za duzo klientów");
                  return;
               } else {
                  activeClients.add(clientId);
                  out.writeObject("OK");
                  System.out.println("Klient " + clientId + " Zaakceptowano");
               }
            }

            for (int i = 0; i < 3; i++) {
               String request = (String) in.readObject();
               String objectType = request.split("_")[1];

               List<Serializable> collection = (List<Serializable>)
objectsMap.getOrDefault(objectType, Collections.emptyList());

               if (collection.isEmpty()) {
                  out.writeObject("Brak obiektów");
               } else {
```

```
                out.writeObject(collection);
            }

            System.out.println("Wysłano " + objectType + " kolekcja do klienta " + clientId);
            Thread.sleep((int) (Math.random() * 1000));
          }
        } catch (Exception e){
          e.printStackTrace();
        } finally {
          synchronized (activeClients){
            activeClients.remove(clientId);
          }
          try {
            clientSocket.close();
          } catch (IOException e){
            e.printStackTrace();
          }
        }

      }
    }
}
```


Ryba
Serwer
External Libraries
11 requestObjects(clientId, objType: "Kot");
12 requestObjects(clientId, objType: "Pies");
Serwer
Klient
Klient 2 Otrzymano Pies Kolekcja: [Pies{rasa: 'Maltańczyk'}, Pies{rasa: 'Labrador'}, Pies{rasa: 'Mops'}, Pies{rasa: 'Golden Re
Klient 1 Otrzymano Pies Kolekcja: [Pies{rasa: 'Maltańczyk'}, Pies{rasa: 'Labrador'}, Pies{rasa: 'Mops'}, Pies{rasa: 'Golden Re
Klient 1: Połączenie zaakceptowane
Klient 1 Otrzymano Ryba Kolekcja: [Ryba{gatunek: 'Okoń'}, Ryba{gatunek: 'Sandacz'}, Ryba{gatunek: 'Sum'}, Ryba{gatunek: 'Karp'
Klient 5 Otrzymano Pies Kolekcja: [Pies{rasa: 'Maltańczyk'}, Pies{rasa: 'Labrador'}, Pies{rasa: 'Mops'}, Pies{rasa: 'Golden Re
Klient 5: Połączenie zaakceptowane
Klient 5 Otrzymano Ryba Kolekcja: [Ryba{gatunek: 'Okoń'}, Ryba{gatunek: 'Sandacz'}, Ryba{gatunek: 'Sum'}, Ryba{gatunek: 'Karp'}
Klient 2 Otrzymano Pies Kolekcja: [Pies{rasa: 'Maltańczyk'}, Pies{rasa: 'Labrador'}, Pies{rasa: 'Mops'}, Pies{rasa: 'Golden Ret
Klient 2: Połączenie zaakceptowane
Klient 2 Otrzymano Ryba Kolekcja: [Ryba{gatunek: 'Okoń'}, Ryba{gatunek: 'Sandacz'}, Ryba{gatunek: 'Sum'}, Ryba{gatunek: 'Karp'}
Klient 2 Otrzymano Ryba Kolekcja: [Ryba{gatunek: 'Okoń'}, Ryba{gatunek: 'Sandacz'}, Ryba{gatunek: 'Sum'}, Ryba{gatunek: 'Karp'}
Klient 2 Otrzymano Ryba Kolekcja: [Ryba{gatunek: 'Okoń'}, Ryba{gatunek: 'Sandacz'}, Ryba{gatunek: 'Sum'}, Ryba{gatunek: 'Karp'}
Klient 1 Otrzymano Ryba Kolekcja: [Ryba{gatunek: 'Okoń'}, Ryba{gatunek: 'Sandacz'}, Ryba{gatunek: 'Sum'}, Ryba{gatunek: 'Karp'}
Klient 5 Otrzymano Ryba Kolekcja: [Ryba{gatunek: 'Okoń'}, Ryba{gatunek: 'Sandacz'}, Ryba{gatunek: 'Sum'}, Ryba{gatunek: 'Karp'}
Klient 1 Otrzymano Ryba Kolekcja: [Ryba{gatunek: 'Okoń'}, Ryba{gatunek: 'Sandacz'}, Ryba{gatunek: 'Sum'}, Ryba{gatunek: 'Karp'}
Klient 5 Otrzymano Ryba Kolekcja: [Ryba{gatunek: 'Okoń'}, Ryba{gatunek: 'Sandacz'}, Ryba{gatunek: 'Sum'}, Ryba{gatunek: 'Karp'}
Process finished with exit code 0