Politechnika Świętokrzyska w Kielcach	
Programowanie obiektowe (Java)	Projekt
Grupa 11B	Paweł Piotrowski Igor Ślusarski

1. Wstęp

Celem projektu było stworzenie gry domino w języku Java z wykorzystaniem architektury server-client. Gra została utworzona i jest funkcjonalna, jednak nie udało się zaimplementować serwera i klienta, przez co nie ma możliwości gry w dwóch osobnych oknach. Program tworzy dwóch graczy, którzy mogą zagrać między sobą, ale tylko w jednym oknie. W repozytorium znajdują się pliki dotyczące serwera i klienta, nie zostały jednak wykorzystane podczas tworzenia pliku .jar, za pomocą którego uruchamiany jest program.

2. Założenia

- Dwóch graczy
- Pula kostek – 28
- Każdy z graczy otrzymuje po 7 kostek
- Grę rozpoczyna Gracz1
- Pierwsza kostka na planszy jest losowana z kostek pozostałych po rozdaniu
- Pierwszy z graczy, który wyłoży wszystkie swoje kostki wygrywa
- W przypadku braku możliwości wyłożenia kostki, gracz dobiera kostkę z puli kostek do dobrania, jeśli jest pusta, wygrywa przeciwnik
- Jeśli jest możliwość dołożenia kostki z obu stron szeregu znajdującego się na planszy, gra automatycznie dołoży kostkę na prawym końcu

3. Przebieg gry

Każdy z dwóch graczy na początku rozgrywki otrzymuje losowo 7 kostek z całej puli. Następnie z pozostałych kostek losowana jest jedna, która będzie kostką początkową i zostaje wyłożona na planszę. Reszta kostek stanowi pulę kostek do dobierania. Grę rozpoczyna Gracz1. Każdy z graczy ma możliwość wyłożyć kostkę (poprzez kliknięcie jej) ze swojego zestawu na planszę (jeśli można ją dopasować ilością oczek do kostek znajdujących się na obu końcach szeregu umieszczonego na planszy) lub dobrać kostkę (poprzez kliknięcie przycisku dobierz). W przypadku dobrania, gracz otrzymuje do swojego zestawu losową kostkę z puli kostek do dobierania. Gdy któryś z graczy wyłoży wszystkie swoje kostki, gracz ten zwycięża, wyświetla się odpowiedni komunikat i program kończy swoje działanie. Podczas dobierania może okazać się, że pula kostek do dobierania jest pusta. W takim wypadku gracz, który nie może wyłożyć żadnej ze swoich kostek na planszę i musi dobrać kostkę, przegrywa, a zwycięzcą zostaje przeciwnik.

4. Zawartość repozytorium

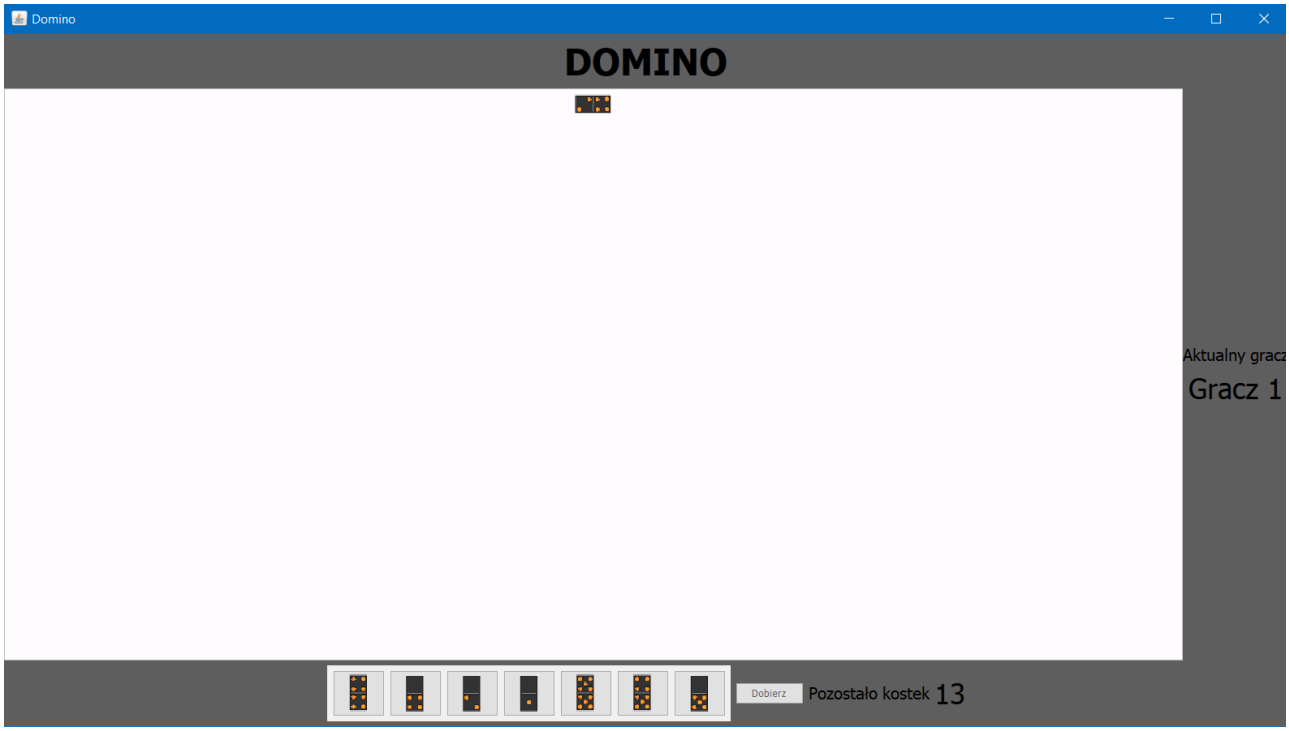
W folderze src znajdują się pliki:

- Pliki Gui.form i Gui.java – zawierają informacje o wyglądzie interfejsu, układzie okien, napisów i przycisków
- Plik Gra.java – zawiera informacje oraz funkcje dotyczące rozgrywki
- Plik Gracz.java – zawiera informacje oraz funkcje dotyczące graczy oraz ich zestawów
- Plik Kostki.java – zawiera informacje oraz funkcje dotyczące puli kostek
- Plik RotatedIcon.java – zawiera funkcje obracające kostkami podczas rozgrywki
- Plik Main.java – główny plik programu, tworzone są w nim obiekty klas Gui oraz Gra

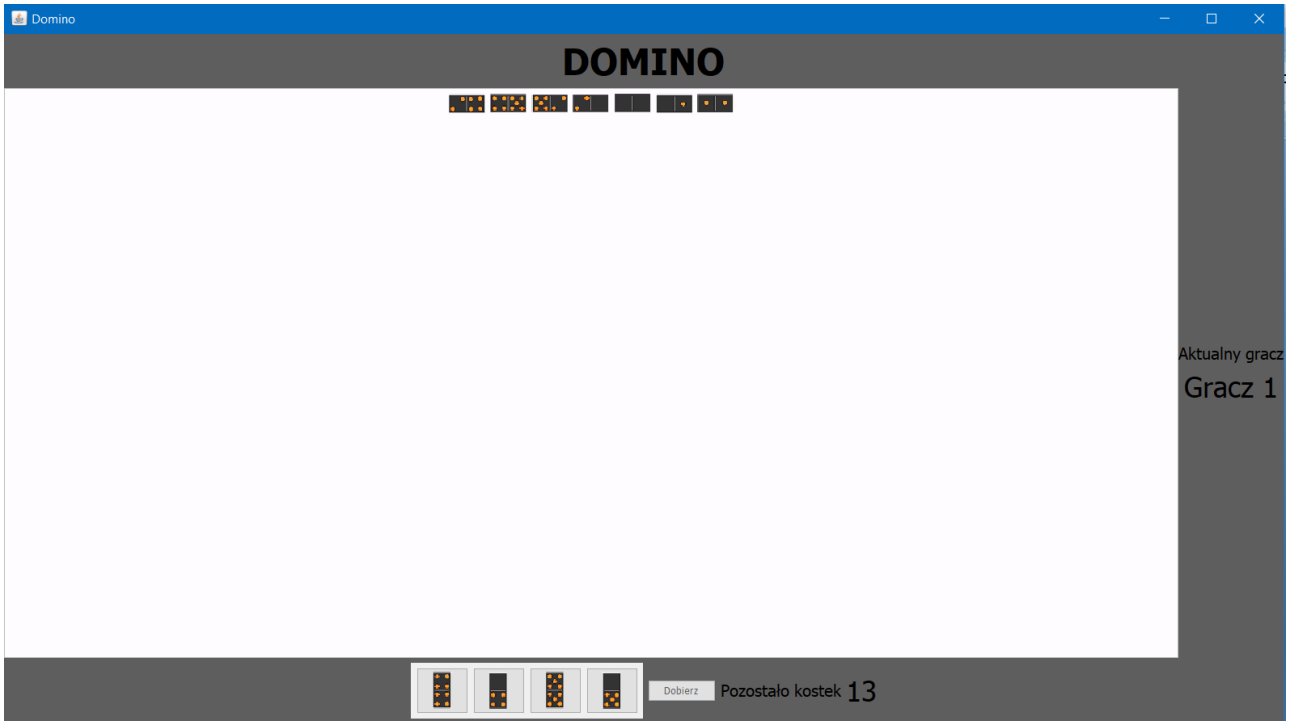
W repozytorium znadują się także pliki Serwer.java, Klient.java oraz Komunikacja.java. Są to pliki dotyczące serwera i klienta. Nie były uwzględnione podczas kompilacji.

W głównym katalogu mieści się również plik .jar, za pomocą którego uruchamiany jest program.

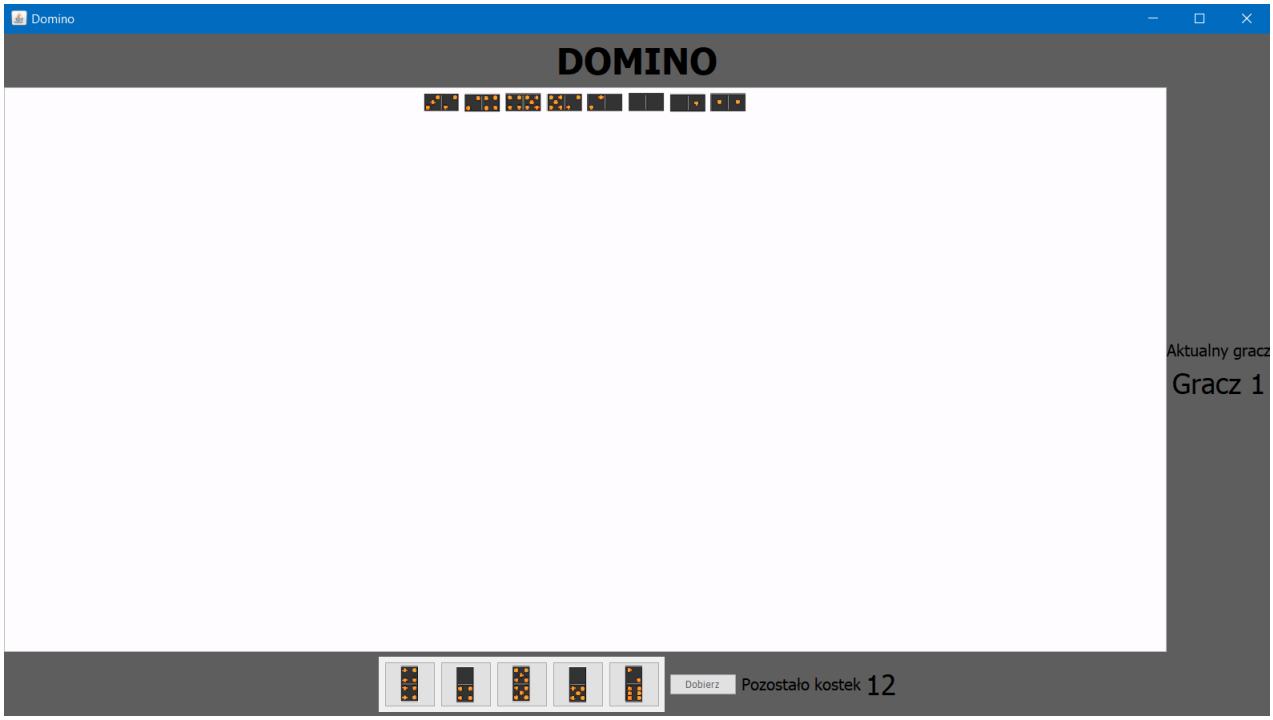
5. Przykłady obrazujące działanie programu



Widok po uruchomieniu – początek rozgrywki

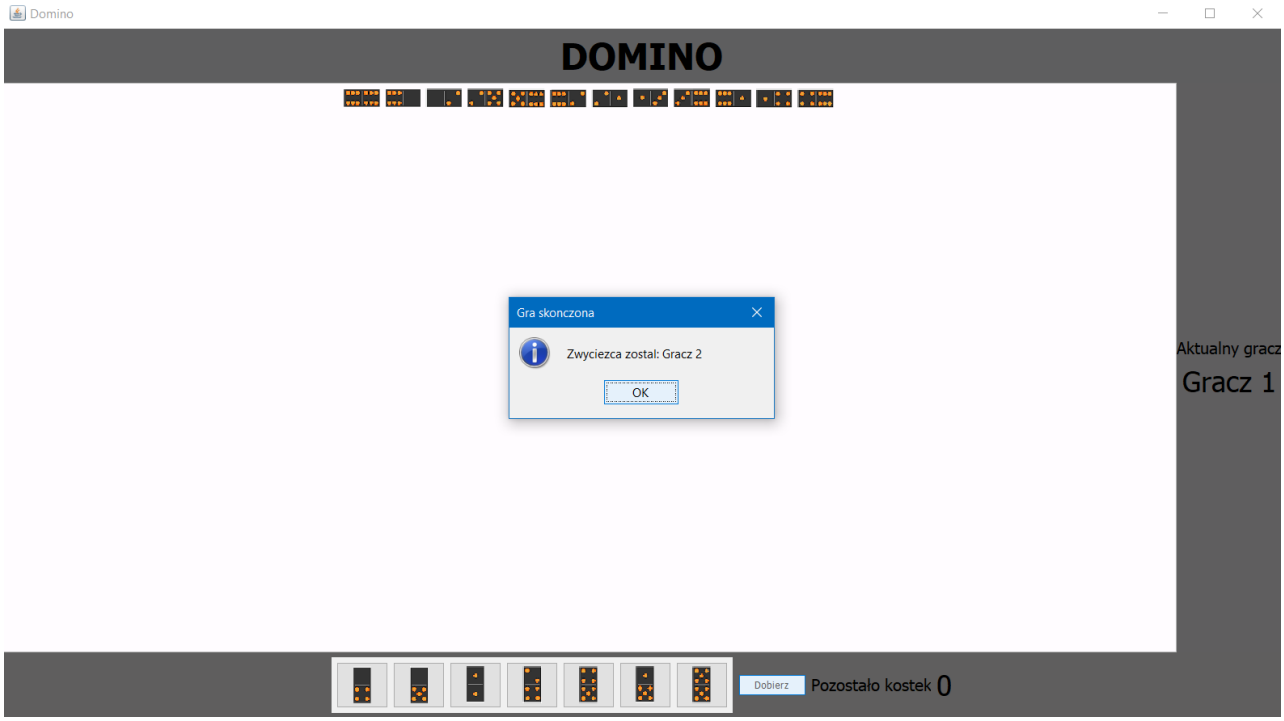


Sytuacja, w której Gracz1 nie może wyłożyć żadnej ze swoich kostek na planszę (wybiera opcję dobierz)



Ruch wykonał Gracz2 i następuje powrót do Gracza1, który dobrał kostkę (liczba kostek do dobrania zmalała o 1, a nowa kostka pojawiła się w zestawie gracza)

Gracz2 wyłożył wszystkie swoje kostki, pojawia się komunikat o wygranej Gracza2



Gracz1 nie może wyłożyć żadnej kostki, pula kostek do dobrania jest pusta; po wciśnięciu przycisku dobierz pojawia się komunikat o wygranej Gracza2

6. Najważniejsze fragmenty kodu

```
public void wylosuj_zestawy(Gracz gracz0, Gracz gracz1){
    ArrayList<String> zestaw_gracza1 = new ArrayList<String>();
    ArrayList<String> zestaw_gracza2 = new ArrayList<String>();

    for(int i = 0; i < 14; i++){
        int id = rand.nextInt(kostki_do_dobierania.size());
        String kostka = kostki_do_dobierania.get(id);
        kostki_do_dobierania.remove(id);
        if(i < 7){
            zestaw_gracza1.add(kostka);
        } else {
            zestaw_gracza2.add(kostka);
        }
    }

    gracz0.nadaj_zestaw_kostek(zestaw_gracza1);
    gracz1.nadaj_zestaw_kostek(zestaw_gracza2);
}
```

Fragment kodu przedstawiający funkcję losującą zestawy dla dwóch graczy. W pętli for jest losowane id kostki, która jest uzyskiwana z puli wszystkich kostek (przed rozpoczęciem rozgrywki pula kostek do dobierania stanowi pulę wszystkich kostek), a następnie pierwsze 7 jest przydzielane do pierwszego gracza, a pozostałe 7 do gracza 2. (fragment znajduje się w pliku Kostki.java)

```
public boolean wykonajAkcje(Gracz gracz) {
        gracz.odswiez_zestaw();
        String decyzja = gui.zagraj();
        if (decyzja.toUpperCase().equals("DOBIERZ")) {
            return dobieranie(gracz);
        } else {
            if (gracz.sprawdz_kostki().contains(decyzja)) {
                if(!sprawdzDecyzje(gracz, decyzja)) {
                    return wykonajAkcje(gracz);
                } else {
                    gracz.odswiez_zestaw();
                    return true;
                }
            } else {
                return wykonajAkcje(gracz);
            }
        }
    }
```

Fragment kodu odpowiedzialny za wykonanie akcji przez gracza. Gracz może dobrać kostkę, wtedy wykona się kod odpowiedzialny za dobieranie lub spróbować wyłożyć którąś kostkę ze swojego zestawu, wtedy zostanie sprawdzone, czy kostka pasuje do szeregu znajdującego się na planszy. (fragment znajduje się w pliku Gra.java)

```
public boolean dobieranie(Gracz gracz){
    if (kostki.sprawdz_ilosc_kostek_do_dobierania() == 0) {
        if(gracz.nick.equals("Gracz 1")){
            zwyciezca = gracz1;
        } else {
            zwyciezca = gracz0;
        }
        koniec = true;
    } else {
        kostki.dobierz_kostke(gracz);
        gui.ustawKostkiPozostale(kostki.sprawdz_ilosc_kostek_do_dobierania());
            System.out.println(gracz.pokaz_kostki());
    }
        gracz.odswiez_zestaw();
    return true;
}
```

Fragment kodu odpowiedzialny za dobieranie kostki przez gracza. Znajduje się tutaj również sprawdzenie, czy pula kostek do dobierania jest pusta. Jeśli warunek jest spełniony, jest wybierany zwycięzca (opis takiego zakończenia rozgrywki znajduje się w przedostatnim punkcie w założeniach). W przeciwnym razie jest wywoływana funkcja dobierz_kostke i liczba kostek do dobrania na ekranie zostaje zaktualizowana. (fragment znajduje się w pliku Gra.java)

```
public void dobierz_kostke(Gracz gracz){
    int id = rand.nextInt(kostki_do_dobierania.size());
    String kostka = kostki_do_dobierania.get(id);
    kostki_do_dobierania.remove(id);
    gracz.dodaj_kostke(kostka);
}
```

Fragment kodu odpowiedzialny za dobranie kostki przez gracza. Losowane jest id kostki z puli kostek do dobierania i jest ona dodawana do zestawu gracza oraz usuwana z puli kostek do dobierania. (fragment znajduje się w pliku Kostki.java)

```java
public boolean sprawdzDecyzje(Gracz gracz, String decyzja){
    if (szereg.endsWith(String.valueOf(decyzja.charAt(0)))) {
        szereg = szereg + " | " + decyzja;
        gracz.usun_kostke_z_zestawu(decyzja);
        gui.polozDomino(decyzja, "r", false);
        return true;
    } else if (szereg.endsWith(String.valueOf(decyzja.charAt(2)))) {
        szereg = szereg + " | " + new StringBuilder(decyzja).reverse().toString();
        gracz.usun_kostke_z_zestawu(decyzja);
        gui.polozDomino(decyzja, "r", true);
        return true;
    } else if (szereg.startsWith(String.valueOf(decyzja.charAt(0)))) {
        szereg = new StringBuilder(decyzja).reverse().toString() + " | " + szereg;
        gracz.usun_kostke_z_zestawu(decyzja);
        gui.polozDomino(decyzja, "f", true);
        return true;
    } else if (szereg.startsWith(String.valueOf(decyzja.charAt(2)))) {
        szereg = decyzja + " | " + szereg;
        gracz.usun_kostke_z_zestawu(decyzja);
        gui.polozDomino(decyzja, "f", false);
        return true;
    } else {
        return false;
    }
}
```

Fragment kodu odpowiedzialny za sprawdzenie, czy kostka może zostać wyłożona na planszę. Każda z kostek ma formę np. „2-5”. Jest to kostka, która ma 2 oraz 5 oczek. Porównywane są pozycje 0 oraz 2 takiej kostki z pozycjami 0 oraz 2 kostek znajdujących się na planszy. Jest kilka możliwości, np. oczka z prawej strony kostki równają się ilości oczek na początku szeregu znajdującego się na planszy, więc kostka może zostać tam dołożona. Wtedy kostka jest usuwana z zestawu gracza i dokładana na planszę (możliwe jest jej obrócenie zależne od miejsca położenia oraz zgodności oczek z jednej bądź drugiej strony kostki). (fragment znajduje się w pliku Gra.java)

```
public void polozDomino(String kostka, String miejsce, boolean odwroc){
    ImageIcon image = new ImageIcon(this.getClass().getResource("/img/" + kostka + ".png"));
    RotatedIcon ri = new RotatedIcon(image, RotatedIcon.Rotate.UPSIDE_DOWN);
    JLabel picLabel = new JLabel();
    if(odwroc) {
        picLabel.setIcon(ri);
    } else {
        picLabel.setIcon(image);
    }
    if(miejsce.equals("r")) {
        jpanelCenter.add(picLabel);
    } else if (miejsce.equals("f")){
        jpanelCenter.add(picLabel, 0);
    }
    revalidate();
    repaint();
}
```

Fragment kodu odpowiedzialny za położenie kostki na planszy. Kostka może zostać położona z dwóch stron szeregu znajdującego się na planszy, może zostać również odwrócona (wywoływana jest w poprzedniej funkcji (sprawdzającej czy kostka może zostać wyłożona na planszę)). (fragment znajduje się w pliku Gui.java)

```
public String wylosuj_kostke_poczatkowa(){
    int id = rand.nextInt(kostki_do_dobierania.size());
    String kostka = kostki_do_dobierania.get(id);
    kostki_do_dobierania.remove(id);
    return kostka;
}
```

Fragment kodu odpowiedzialny za losowanie kostki początkowej, czyli takiej, która na początku gry zostanie wyłożona na stół i to od niej rozpocznie się właściwa rozgrywka. Losowane jest id kostki z puli kostek do dobierania i jest ona usuwana z tej puli. (fragment znajduje się w pliku Kostki.java)

```
String poczatkowaKostka = kostki.wylosuj_kostke_poczatkowa();
this.gui.polozDomino(poczatkowaKostka, "f", false);
```

Linijki odpowiedzialne za położenie na planszy wylosowanej kostki początkowej (fragment znajduje się w pliku Gra.java).

```
while (!koniec) {
    if (aktu == 0) {
        ruch(gracz0);
        aktu = 1;
    } else if (aktu == 1){
        ruch(gracz1);
        aktu = 0;
    }
    if(gracz0.ilosc_kostek() == 0) {
        zwyciezca = gracz0;
        koniec = true;
    } else if (gracz1.ilosc_kostek() == 0) {
        zwyciezca = gracz1;
        koniec = true;
    }
}
```

Fragment kodu odpowiedzialny za wymianę ruchów pomiędzy graczami oraz kończenie rozgrywki w przypadku wyłożenia wszystkich kostek z zestawu u jednego z graczy.

```
public boolean dodajKlienta(Klient k) {
    k.tell("Oczekiwanie na polaczenie");
    for (Klient klient : Klienci) {
        klient.tell("Pojawil sie nowy klient");
    }
    k.gracz = new Gracz();
    Klienci.add(k);
    System.out.printf("Dodano klienta");
    return true;
}

public static Klient przyjmijKlienta(){
    Klient klient = null;
    try{
        klient = new Klient(socket.accept());
        System.out.println("[Server] Klient przyjety");
    } catch (IOException e) {
        System.out.println("[Server] Blad przyjecia klienta");
    }
    return klient;
}
```

Fragment kodu odpowiedzialny za przyjmowanie i dodawanie klientów. Pochodzi z pliku Serwer.java. Plik nie został uwzględniony podczas kompilacji (bardziej szczegółowy opis znajduje się we wstępie).

7. Wnioski

Tworzenie powyższego programu pozwoliło nam na zapoznanie się oraz utrwalenie wiedzy o programowaniu obiektowym w języku Java.