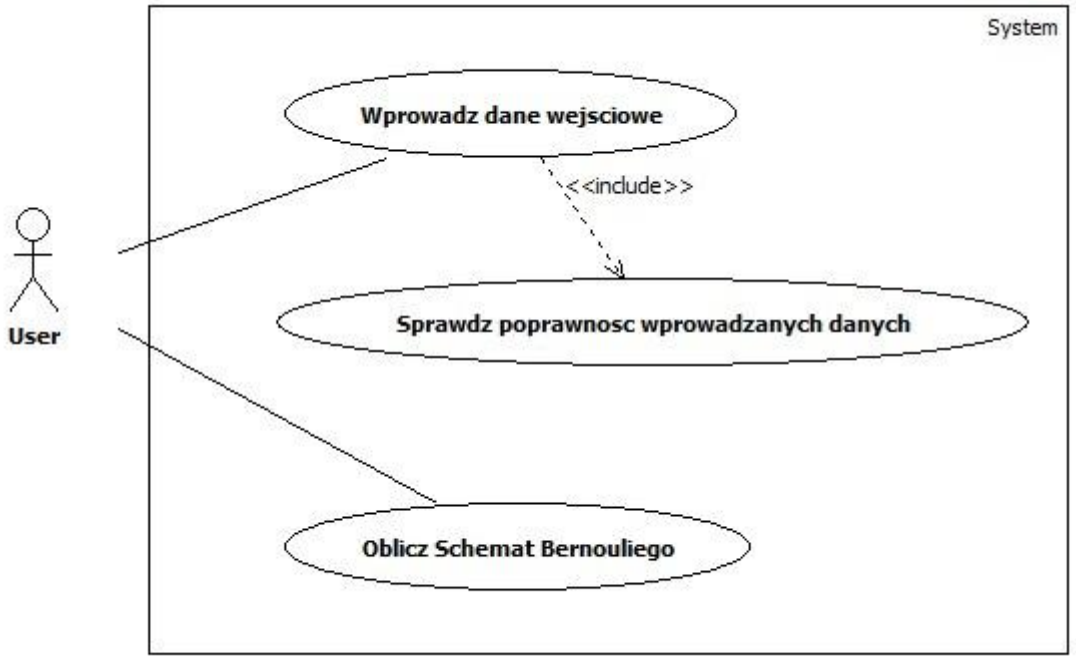
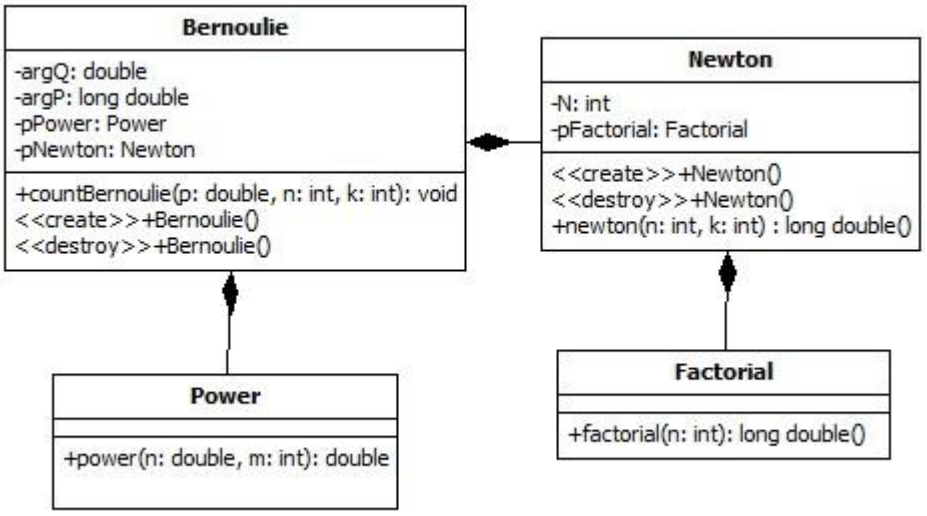
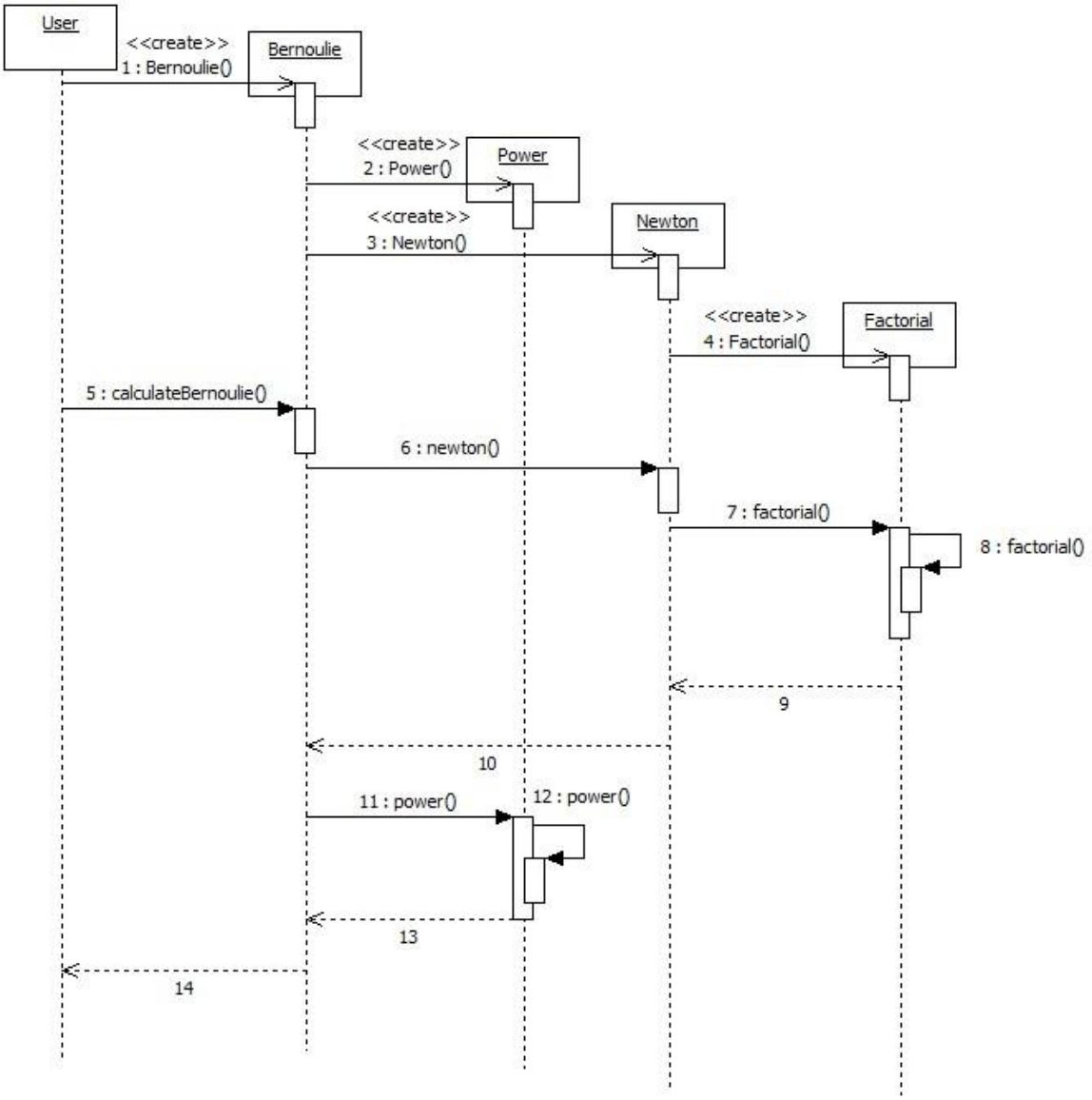
1. Diagram przypadków użycia



2. Model logiczny problemu



3. Diagram sekwencji



4. Implementacja systemu

```h
#if !defined(_POWER_H)
#define _POWER_H


class Power {
public:
    double power(double n, int m);
};

#endif  //_POWER_H
```

```cpp
#include "Power.h"

double Power::power(double n, int m) {
    return m==0 ? 1.0 : n * power(n,m-1);
}
```

```h
#if !defined(_FACTORIAL_H)
#define _FACTORIAL_H


class Factorial {
public:
    long double factorial(int n);
};

#endif  //_FACTORIAL_H
```

```cpp
#include "Factorial.h"

long double Factorial::factorial(int n) {
    return n==0 ? 1.0 : n * factorial(n-1);
}
```

```
#if !defined(_NEWTON_H)
#define _NEWTON_H

#include "Factorial.h"

class Newton {
public:
    Newton();
    ~Newton();
    long double newton(int n,int k);
private:
    int N;
    Factorial *pFactorial;
};

#endif  //_NEWTON_H
```

```cpp
#include "Newton.h"

Newton::Newton() {
    pFactorial = new Factorial();
}

Newton::~Newton() {
    delete pFactorial;
}

long double Newton::newton(int n, int k){
    N = n-k;
    N = n - k;
    long NbyK = 1;
    if(k >= N) {
        for(int i = k+1; i <= n; i++)
            NbyK *= i;
        return (NbyK/pFactorial->factorial(n - k));
    }
    else {
        for(int i = N+1; i <= n; i++)
            NbyK *= i;
        return (NbyK / pFactorial->factorial(k));
    }
}
```

```cpp
#if !defined(_BERNOULIE_H)
#define _BERNOULIE_H

#include "Power.h"
#include "Newton.h"
#include <iostream>

using namespace std;

class Bernoulie {
public:
    void countBernoulie(double p, int n, int k);
    Bernoulie();
    ~Bernoulie();
private:
    double argQ;
    long double argP;
    Power *pPower;
    Newton *pNewton;
};

#endif  //_BERNOULIE_H
```

```cpp
#include "Bernoulie.h"

void Bernoulie::countBernoulie(double p, int n, int k) {
    argQ = 1 - p;
    auto result = (pNewton->newton(n,k) *
            pPower->power(p,k) *
            pPower->power(argQ,n-k));
    cout << "Prawdopodobienstwo, ze w " << n << " doswiadczeniach "
        << k << " razy otrzymamy sukces wynosi " << result <<endl;

}

Bernoulie::Bernoulie() {
    pNewton = new Newton();
    pPower = new Power();
}

Bernoulie::~Bernoulie() {
    delete pNewton;
    delete pPower;
}
```

```cpp
#include "Bernoulie.h"

int main() {
    double p;
    int n, k;
    cout << " Podaj p: ";
    cin >> p;
    cout << " Podaj n: ";
    cin >> n;
    cout << " Podaj k: ";
    cin >> k;
    Bernoulie *bernoulie = new Bernoulie();
    bernoulie->countBernoulie(p,n,k);
    return 0;
}
```

5. Test systemu

```
Podaj p: 0.5
Podaj n: 6
Podaj k: 2
Prawdopodobienstwo, ze w 6 doswiadczeniach 2 razy otrzymamy sukces wynosi 0.234375
```