```
class Stack {
private:
    int* arr;
    int top;
    int capacity;

    static int objectCount;
public:
    Stack() {
        capacity = 5;
        arr = new int[capacity];
        top = -1;
        objectCount++;
        cout << "Default constructor .\n";
    }

    Stack(int size) {
        capacity = size;
```

```
    arr = new int[capacity];
    top = -1;
    objectCount++;
    cout << "Parameterized constructor .\n";
}

Stack& push(int value) {
    if (top == capacity - 1) {
        cout << "Stack Overflow! Cannot push " << value << endl;
    } else {
        arr[++top] = value;
    }
    return *this;
}

Stack& pop() {
    if (top == -1) {
        cout << "Stack Underflow! Nothing to pop.\n";
```

```
        return *this;
    }

    void showElements() const {
        if (top == -1) {
            cout << "Stack is empty.\n";
            return;
        }
        cout << "Stack elements: ";
        for (int i = 0; i <= top; ++i) {
            cout << arr[i] << " ";
        }
        cout << endl;
    }

    static void showObjectCount() {
        cout << "Number of Stack objects created: " << objectCount << endl;
    }
```

```cpp
int Stack::objectCount = 0;

int main() {

    Stack s1;
    s1.push(10).push(20).push(30);
    s1.showElements();
    s1.pop().showElements();

    Stack s2(3);
    s2.push(5).push(15).push(25);
    s2.showElements();

    Stack::showObjectCount();

    return 0;
}
```

```cpp
class Complex {
private:
    float real;
    float imag;

    static int counter;

public:
    Complex() {
        real = 0;
        imag = 0;
        counter++;
        cout << "Default constructor called." << endl;
    }

    Complex(float r) {
        real = r;
        imag = 0;
        counter++;
        cout << "Overloaded constructor (1 parameter) called." << endl;
    }
```

```
    Complex(float r, float i) {
        real = r;
        imag = i;
        counter++;
        cout << "Overloaded constructor (2 parameters) called." << endl;
    }

    ~Complex() {
        counter--;
        cout << "Destructor called for object (" << real << ", " << imag << "i)" << endl;
    }

    static void showCounter() {
        cout << "Number of active Complex objects: " << counter << endl;
    }

    void printComplex() const {
        if (real == 0 && imag == 0)
            cout << "0";
        else if (real == 0)
            cout << imag << "i";
```

```
    void printComplex() const {
        if (real == 0 && imag == 0)
            cout << "0";
        else if (real == 0)
            cout << imag << "i";
        else if (imag == 0)
            cout << real;
        else if (imag == 1)
            cout << real << "+i";
        else if (imag == -1)
            cout << real << "-i";
        else if (imag > 0)
            cout << real << "+" << imag << "i";
        else
            cout << real << imag << "i";
        cout << endl;
    }
};

int Complex::counter = 0;
```

```cpp
int main() {
    cout << "--- Creating objects ---" << endl;

    Complex c1;
    Complex c2(5);
    Complex c3(5, 7);
    Complex c4(-3, -3);
    Complex c5(0, -8);
    Complex c6(0, 8);
    Complex c7(-9, 0);
    Complex c8(-9, 3);
    Complex c9(8, 1);
    Complex c10(8, -1);

    cout << "\n--- Displaying complex numbers ---" << endl;
    c1.printComplex();
    c2.printComplex();
    c3.printComplex();
    c4.printComplex();
    c5.printComplex();
```

```
cout << "\n--- Displaying complex numbers ---" << endl;
c1.printComplex();
c2.printComplex();
c3.printComplex();
c4.printComplex();
c5.printComplex();
c6.printComplex();
c7.printComplex();
c8.printComplex();
c9.printComplex();
c10.printComplex();

cout << "\n--- Showing object counter ---" << endl;
Complex::showCounter();

return 0;
```

```cpp
class BankAccount {
private:
    string accountHolder;
    int accountNumber;
    double balance=0;

    static int counter;
    static int nextID;

public:
    BankAccount() = default;

    BankAccount(const string& name, double initialBalance = 0.0) {
        accountHolder = name;
        accountNumber = ++nextID;
        balance = initialBalance;
        counter++;
```

```
}

BankAccount& deposit(double amount) {
    if (amount > 0) {
        balance += amount;
        cout << "Deposited: " << amount << " to Account No: " << accountNumber << endl
    } else {
        cout << "Invalid deposit amount.\n";
    }
    return *this;
}

void withdraw(double amount) {
    if (amount <= 0) {
        cout << "Invalid withdrawal amount.\n";
    } else if (amount > balance) {
```

```
    double getBalance() const {
        return balance;
    }

    static void showCounter() {
        cout << "Total Bank Accounts Created: " << counter << endl;
    }

    ~BankAccount() {
        cout << "Account [" << accountNumber << "] of " << accountHolder << " closed.\n";
    }
;

int BankAccount::counter = 0;
int BankAccount::nextID = 1000;
```

```
BankAccount acc0;
BankAccount acc1("Ahmed Essam", 1000);
BankAccount acc2("Sara Ali", 5000);

acc1.deposit(500).deposit(250).withdraw(300);
acc2.deposit(1000).withdraw(700);

cout << "\n--- Account Details ---\n";
acc0.showAccount();
acc1.showAccount();
acc2.showAccount();

cout << "\n--- Checking Balances ---\n";
cout << acc1.getBalance() << " EGP" << endl;
cout << acc2.getBalance() << " EGP" << endl;

cout << "\n--- Total Accounts ---\n";
BankAccount::showCounter();
```