

A dark blue vertical bar on the left side of the slide. A blue arrow points to the right from the bar, containing the date.

3/12/2025

Object Oriented Programming

Lab 6

Several thin, curved lines in dark blue and light grey originate from the bottom left corner and sweep upwards and to the right.

Mohammad Arham Usman
FAST NUCES

Question 1

Source Code:

```
#include <iostream>
#include <string>
using namespace std;

class Message {
    string content;
public:
    //Constructor
    Message(string Content="") {
        content = Content;
    }

    //Setter
    void setContent(const string Content) {
        content = Content;
    }

    //Getter
    string getContent() const {
        return content;
    }

    //Method
    virtual string toString() {
        return content;
    }
};

class KeywordSearchAndEncoding {
public:
    bool containsKeyword(const Message& messageObject, const string& keyword) {
        string content = messageObject.getContent();
        bool found = false;
        int length = content.length();
        for (int i = 0; i < length; i++) {
            if (content.at(i) == keyword.at(0)) {
                found = true;
                for (int j = 0; j < keyword.length(); j++) {
                    if ((i + j) == length) {
                        found = false;
                        break;
                    }
                    if (content.at(i + j) != keyword.at(j)) {
                        found = false;
                        break;
                    }
                }
                if (found) return true;
            }
        }
        return false;
    }
};
```

```

    }
    void encoding(Message& messageObject) {
        string content = messageObject.getContent();
        for (int i = 0; i < content.length(); i++) {
            if (content.at(i) == 'z') {
                content.replace(i, 1, "a");
            }
            if (content.at(i) == 'Z') {
                content.replace(i, 1, "A");
            }
            if ((content.at(i) >= 'a' && content.at(i) < 'z') ||
content.at(i) >= 'A' && content.at(i) < 'Z') {
                string next(1, content.at(i)+1);
                content.replace(i, 1, next);
            }
        }
        messageObject.setContent(content);
    }
    void decoding(Message& messageObject) {
        string content = messageObject.getContent();
        for (int i = 0; i < content.length(); i++) {
            if (content.at(i) == 'a') {
                content.replace(i, 1, "z");
            }
            if (content.at(i) == 'A') {
                content.replace(i, 1, "Z");
            }
            if ((content.at(i) > 'a' && content.at(i) <= 'z') ||
content.at(i) > 'A' && content.at(i) <= 'Z') {
                string next(1, --content.at(i));
                content.replace(i, 1, next);
            }
        }
        messageObject.setContent(content);
    }
};

class Sms :public Message, protected KeywordSearchAndEncoding {
    string recipientContactNo;
    bool validNumber(const string a) const {
        if (a.length() != 11) return false;
        for (int i = 0; i < 11; i++) {
            if (!(a.at(i) >= '0' && a.at(i) <= '9')) return false;
        }
        return true;
    }
public:
    //Constructor
    Sms(string ContactNo="00000000000", string message = ""): Message(message) {
        setContactNo(ContactNo);
    }

    //Setter
    bool setContactNo(const string ContactNo) {
        bool set = validNumber(ContactNo);
        if (set) {
            recipientContactNo = ContactNo;
        }
    }
};

```

```

        else {
            cout << "Error! Invalid Contact No\n";
        }
        return set;
    }

    //Getter
    string getContactNo() const {
        return recipientContactNo;
    }

    //Method
    bool keywordSearch(string keyword) {
        return containsKeyword(*this, keyword);
    }
    void encode() {
        encoding(*this);
    }
    void decode() {
        decoding(*this);
    }
    string toString() override {
        return recipientContactNo + ":@" + getContent();
    }
    void displayDetails() {
        cout << "-----\nSMS:\n";
        cout << "Recipient Contact No: " << recipientContactNo << endl;
        cout << getContent() << endl;
        cout << "-----\n";
    }
};

class Email : public Message, protected KeywordSearchAndEncoding {
    string sender, reciever, subject;
    bool validInput(string a) {
        int length = a.length();
        if (length < 3) return false;
        if (!(a.at(0) > 'a' && a.at(0) < 'z') || (a.at(0) > 'A' && a.at(0) <
'Z')) {
            return false;
        }
        for (int i = 0; i < length; i++) {
            if (a.at(i) == ' ' || a.at(i) == ':' || a.at(i) == ';') return false;
        }
        return true;
    }
public:
    //Constructor
    Email(string sender="", string reciever="", string subject="", string
message="") : Message(message) {
        if (sender != "") setSender(sender);
        else this->sender = "";
        if (reciever != "") setReciever(reciever);
        else this->reciever = "";
        if (subject != "") setSubject(subject);
        else this->subject = "";
    }
}

```

```

//Setter
bool setSender(const string Sender) {
    bool valid = validInput(Sender);
    if (valid) sender = Sender;
    else cout << "Invalid Sender\n";
    return valid;
}
bool setReciever(string Reciever) {
    bool valid = validInput(Reciever);
    if (valid) reciever = Reciever;
    else cout << "Invalid Reciever\n";
    return valid;
}
bool setSubject(string Subject) {
    bool valid = validInput(Subject);
    subject = Subject;
    return valid;
}

//Getter
string getSender() {
    return sender;
}
string getReciever() {
    return reciever;
}
string getSubject() {
    return subject;
}

//Method
bool keywordSearch(string keyword) {
    return containsKeyword(*this, keyword);
}
void encode() {
    encoding(*this);
}
void decode() {
    decoding(*this);
}
string toString() override {
    return sender + "::<" + reciever + "::<" + subject + "::<" + getContent();
}
void displayDetails() {
    cout << "-----\nE-MAIL\n";
    cout << "Sender: " << sender << endl;
    cout << "Reciever: " << reciever << endl;
    cout << "Subject: " << subject << endl;
    cout << "\n" << getContent() << endl;
    cout<< "\n-----\n";
}
};

int main() {
    Sms s1("03313755393", "This is Java");
    Email e1("m.arhamusman17@gmail.com", "shafique.rehman@nu.edu.pk", "Query
about Inheritance", "Assalam Aliakum Sir! Kindly explain the diamond problem");
    cout << "Original sms:\n";

```

```

s1.displayDetails();
cout << "Finding 'Java' in sms: ";
bool found = s1.keywordSearch("Java");
if (found) cout << "Found\n";
else cout << "Not Found\n";
cout << "-----\n";
cout << "Encoding sms ..... \n";
s1.encode();
s1.displayDetails();
cout << "Decoding sms:\n";
s1.decode();
s1.displayDetails();
cout << "-----\n";
cout << "Original Email:\n";
e1.displayDetails();
cout << "Finding Java in email: ";
found = e1.keywordSearch("Java");
if (found) cout << "Found\n";
else cout << "Not Found\n";
cout << "-----\n";
cout << "Encoding email ..... \n";
e1.encode();
e1.displayDetails();
cout << "-----\n";
cout << "Decoding email:\n";
e1.decode();
e1.displayDetails();
}

```

Screen Shot:

```

Original sms:
-----
SMS:
Recipient Contact No: 03313755393
This is Java
-----
Finding 'Java' in sms: Found
-----
Encoding sms .....
-----
SMS:
Recipient Contact No: 03313755393
Uijt jt Kbwb
-----
Decoding sms:
-----
SMS:
Recipient Contact No: 03313755393
This is Java
-----

```

Original Email:

E-MAIL

Sender: m.arhamusman17@gmail.com
Reciever: shafique.rehman@nu.edu.pk
Subject: Query about Inheritance

Assalam Aliakum Sir! Kindly explain the diamond problem

Finding Java in email: Not Found

Encoding email.....

E-MAIL

Sender: m.arhamusman17@gmail.com
Reciever: shafique.rehman@nu.edu.pk
Subject: Query about Inheritance

Bttbmbn Bmjblvn Tjs! Ljoemz fyqmbjo uif ejbnpoe qspcmfn

Decoding email:

E-MAIL

Sender: m.arhamusman17@gmail.com
Reciever: shafique.rehman@nu.edu.pk
Subject: Query about Inheritance

Assalam Aliakum Sir! Kindly explain the diamond problem

E:\C++ Programming\OOP LAB 6\x64\Debug\OOP LAB 6.exe (process 1064)
Press any key to close this window . . .