National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

# Department Of Electrical Engineering and Computer Sciences

**Instructor:** Mehreen Tahir **Date:** December 11, 2023

**Lab Engineer:** Mehwish Kiran **Time:** 10:00am – 12:50pm

# CS 212: Object Oriented Programming

# Lab 13: Polymorphism and Abstract Classes

| **Information** | **Description** |
|---|---|
| **Name:** | Irfa Farooq |
| **CMS ID:** | 412564 |
| **Class:** | BEE-14 |
| **Section:** | D |
| **Tenure:** | Fall 2023 |

## Task 1: Dynamic Binding

### Person.h:

```
#pragma once
#include <iostream>

class Person {

protected:
	std::string personName;
	int age;

public:
	Person();
	virtual void Print() = 0;
	void set_personName(std::string);
	std::string get_personName();
	void set_age(int);
	int get_age();
};
```

### Person.cpp:

```
#include<iostream>
#include"Person.h"

Person::Person() :personName("NULL"), age(0) {};

void Person::set_personName(std::string P) {
	personName = P;
}

std::string Person::get_personName() {
	return personName;
}

void Person::set_age(int A) {
	age = A;
}

int Person::get_age() {
	return age;
}
```

### Patient.h:

```
#pragma once
#include<iostream>
#include"Person.h"

class Patient :public Person {
protected:
	std::string diseaseType, recommendedMedicine;
public:
	Patient();
	void Print();
	void set_diseaseType(std::string);
```

```
        std::string get_diseaseType();
        void set_recommendedMedicine(std::string);
        std::string get_recommendedMedicine();
};
```

**Patient.cpp:**

```
#include<iostream>
#include"Patient.h"

Patient::Patient() :diseaseType("NULL"), recommendedMedicine("NULL") {};

void Patient::set_diseaseType(std::string D) {
        diseaseType = D;
}

std::string Patient::get_diseaseType() {
        return diseaseType;
}

void Patient::set_recommendedMedicine(std::string M) {
        recommendedMedicine = M;
}

std::string Patient::get_recommendedMedicine() {
        return recommendedMedicine;
}

void Patient::Print() {
        std::cout << "Patient Details: " << std::endl;
        std::cout << "Name: " << get_personName() << std::endl;
        std::cout << "Age: " << get_age() << std::endl;
        std::cout << "Disease: " << diseaseType << std::endl;
        std::cout << "Recommended Medicine: " << recommendedMedicine << std::endl;
}
```

**MedicarePatient.h:**

```
#pragma once
#include<iostream>
#include"Patient.h"

class MedicarePatient :public Patient {
public:
        std::string hospital, ward;
        int roomNum;
public:
        MedicarePatient();
        void Print();
};
```

**MedicarePatient.cpp:**

```
#include<iostream>
#include"MedicarePatient.h"

MedicarePatient::MedicarePatient() :hospital("Irfa's Hospital"), ward("NULL"),
roomNum(0) {};
```

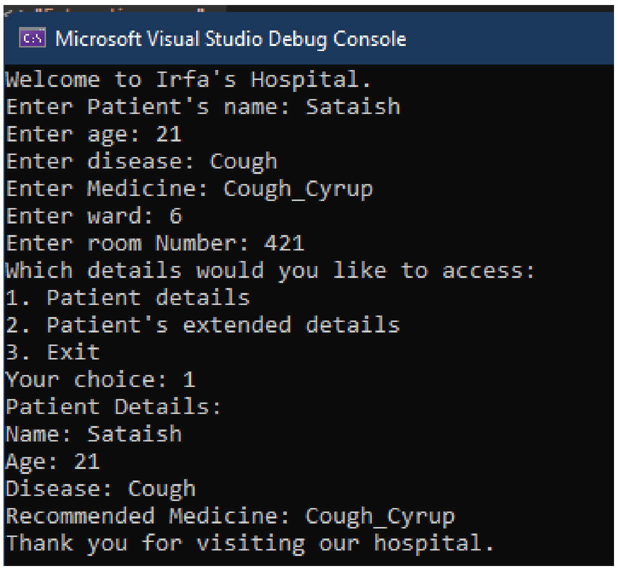
```
void MedicarePatient::Print() {
	std::cout << "Patient extended details: " << std::endl;
	std::cout << "Name: " << get_personName() << std::endl;
	std::cout << "Age: " << get_age() << std::endl;
	std::cout << "Disease: " << get_diseaseType() << std::endl;
	std::cout << "Recommended Medicine: " << get_recommendedMedicine() <<
std::endl;
	std::cout << "Hospital of admittance: " << hospital << std::endl;
	std::cout << "Ward: " << ward << std::endl;
	std::cout << "Room Number: " << roomNum << std::endl;
}
```

**Main.cpp:**

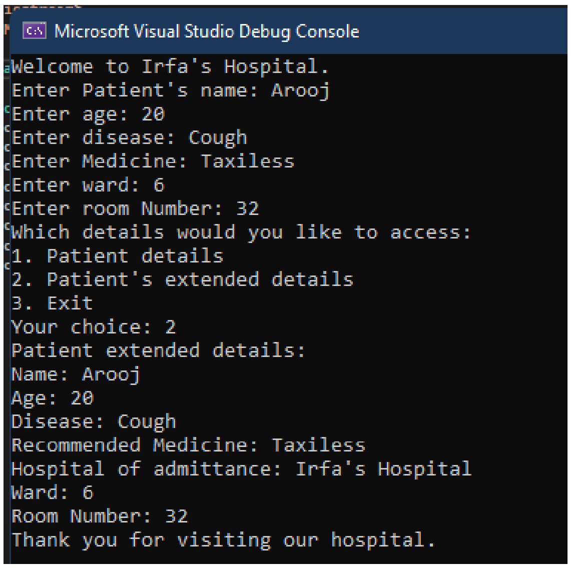
```
#include<iostream>
#include"person.h"
#include"Patient.h"
#include"MedicarePatient.h"

int main() {
	Person* P;
	Patient p;
	MedicarePatient M;
	std::string hospital = "Irfa's Hospital";
	std::string name, disease, medicine;
	int age;
	std::cout << "Welcome to Irfa's Hospital." << std::endl;
	std::cout << "Enter Patient's name: ";
	std::cin >> name;
	std::cout << "Enter age: ";
	std::cin >> age;
	std::cout << "Enter disease: ";
	std::cin >> disease;
	std::cout << "Enter Medicine: ";
	std::cin >> medicine;
	std::cout << "Enter ward: ";
	std::cin >> M.ward;
	std::cout << "Enter room Number: ";
	std::cin >> M.roomNum;
	std::cout << "Which details would you like to access: " << std::endl;
	int choice;
	std::cout << "1. Patient details" << std::endl;
	std::cout << "2. Patient's extended details" << std::endl;
	std::cout << "3. Exit" << std::endl;
	std::cout << "Your choice: ";
	while (1) {
		std::cin >> choice;
		switch (choice) {
		case 1: {
			P = &p;
			p.set_personName(name);
			p.set_age(age);
			p.set_diseaseType(disease);
			p.set_recommendedMedicine(medicine);
			P->Print();
			std::cout << "Thank you for visiting our hospital." <<
std::endl;
			return 0;
		}
```

```
                case 2: {
                        P = &M;
                        M.set_personName(name);
                        M.set_age(age);
                        M.set_diseaseType(disease);
                        M.set_recommendedMedicine(medicine);
                        P->Print();
                        std::cout << "Thank you for visiting our hospital." <<
std::endl;
                        return 0;
                }
                case 3: {
                        std::cout << "Thank you for visiting our hospital." <<
std::endl;
                        return 0;
                }
                default: {
                        std::cout << "Invalid Input!" << std::endl;
                        std::cout << "Input again: ";
                }
                }
        }
}
```

<u>**Output Screenshots**</u>



```
Welcome to Irfa's Hospital.
Enter Patient's name: Sataish
Enter age: 21
Enter disease: Cough
Enter Medicine: Cough_Cyrup
Enter ward: 6
Enter room Number: 421
Which details would you like to access:
1. Patient details
2. Patient's extended details
3. Exit
Your choice: 1
Patient Details:
Name: Sataish
Age: 21
Disease: Cough
Recommended Medicine: Cough_Cyrup
Thank you for visiting our hospital.
```



```
Welcome to Irfa's Hospital.
Enter Patient's name: Arooj
Enter age: 20
Enter disease: Cough
Enter Medicine: Taxiless
Enter ward: 6
Enter room Number: 32
Which details would you like to access:
1. Patient details
2. Patient's extended details
3. Exit
Your choice: 2
Patient extended details:
Name: Arooj
Age: 20
Disease: Cough
Recommended Medicine: Taxiless
Hospital of admittance: Irfa's Hospital
Ward: 6
Room Number: 32
Thank you for visiting our hospital.
```

## Task 2: Inheritance and Polymorphism

### Item.h:

```
#include <iostream>

class Item {
protected:
	std::string title;
	int price;
	int sales[3];
public:
	Item();
	void getData();
	void DisplayData();
};
```

### Item.cpp:

```
#include <iostream>
#include "Item.h"

Item::Item() :title("NULL"), price(0) {
	for (int i = 0; i < 3; i++) {
		*(sales + i) = 0;
	}
}
void Item::getData() {
	title = "Moniter";
	price = 35000;
	*sales = 1200;
	*(sales + 1) = 500;
	*(sales + 2) = 6000;
}
void Item::DisplayData() {
	std::cout << "Item Details: " << std::endl;
	std::cout << "Item Title: " << title << std::endl;
	std::cout << "Item price: " << price << "Rs" << std::endl;
	std::cout << "Sales f{or last three months: " << std::endl;
	for (int i = 0; i < 3; i++) {
		std::cout << "Sales for month " << i + 1 << ": " << *(sales + i) <<
std::endl;
	}
}
```

### HardwareItem.h:

```
#include <iostream>
#include "Item.h"

class HardwareItem :public Item {
protected:
	std::string Manufacturer;
public:
	HardwareItem();
	void getData();
	void DisplayData();
};
```

**HardwareItem.cpp:**

```
#include <iostream>
#include "HardwareItem.h"

HardwareItem::HardwareItem() {
	Manufacturer = "NULL";
}
void HardwareItem::getData() {
	Manufacturer = "Irfa";
	title = "Laptop";
	price = 30000 ;
	*sales = 2200;
	*(sales + 1) = 100;
	*(sales + 2) = 12000;
}
void HardwareItem::DisplayData() {
	std::cout << std::endl;
	std::cout << "Hardware Item Details: " << std::endl;
	std::cout << "Item Title: " << title << std::endl;
	std::cout << "Item Manufacturer: " << Manufacturer << std::endl;
	std::cout << "Item price: " << price << "Rs" << std::endl;
	std::cout << "Sales f{or last three months: " << std::endl;
	for (int i = 0; i < 3; i++) {
		std::cout << "Sales for month " << i + 1 << ": " << *(sales + i) <<
std::endl;
	}
}
```

**SoftwareItem.h:**

```
#include <iostream>
#include "Item.h"

class SoftwareItem :public Item {
protected:
	std::string OperatingSystem;
public:
	SoftwareItem();
	void getData();
	void DisplayData();
};
```

**SoftwareItem.cpp:**

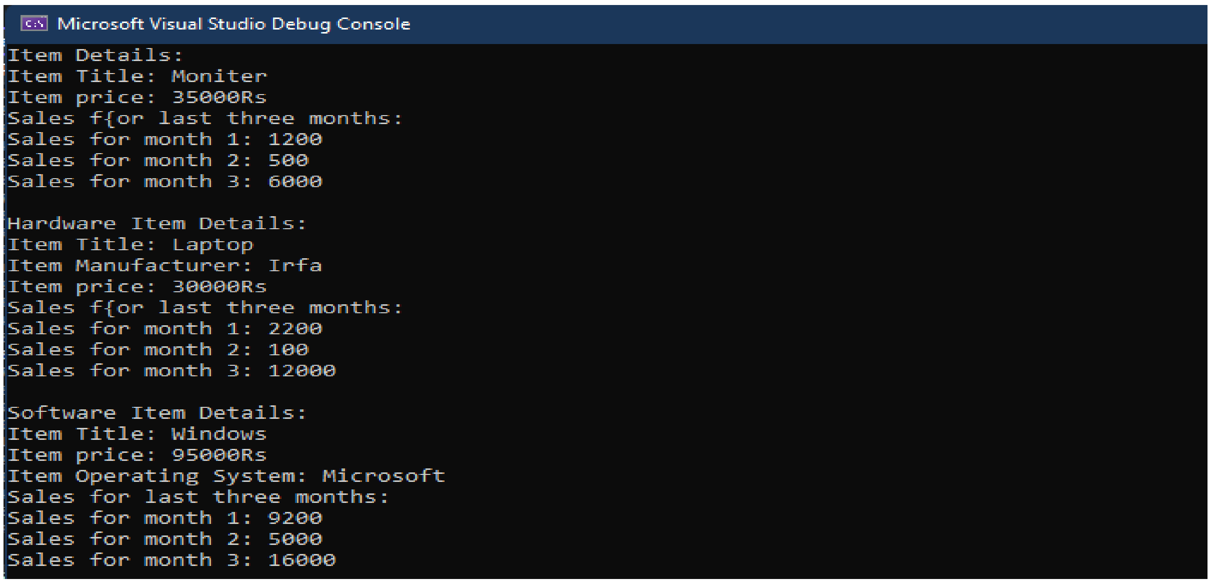
```
#include <iostream>
#include "SoftwareItem.h"

SoftwareItem::SoftwareItem() {
	OperatingSystem = "NULL";
}

void SoftwareItem::getData() {
	title = "Windows";
	price = 95000;
	*sales = 9200;
	*(sales + 1) = 5000;
	*(sales + 2) = 16000;
	OperatingSystem = "Microsoft";
}
```

```
void SoftwareItem::DisplayData() {
	std::cout << std::endl;
	std::cout << "Software Item Details: " << std::endl;
	std::cout << "Item Title: " << title << std::endl;
	std::cout << "Item price: " << price << "Rs" << std::endl;
	std::cout << "Item Operating System: " << OperatingSystem << std::endl;
	std::cout << "Sales for last three months: " << std::endl;
	for (int i = 0; i < 3; i++) {
		std::cout << "Sales for month " << i + 1 << ": " << *(sales + i) <<
std::endl;
	}
}
```

**Main.cpp:**

```
#include <iostream>
#include "Item.h"
#include "HardwareItem.h"
#include "SoftwareItem.h"

void main(){
	Item* it = new Item;
	HardwareItem h1;
	SoftwareItem* s = new SoftwareItem;
	it->getData();
	it->DisplayData();
	h1.getData();
	h1.DisplayData();
	s->getData();
	s->DisplayData();
	delete it;
}
```

**<u>Output Screenshots</u>**



```
Item Details:
Item Title: Moniter
Item price: 35000Rs
Sales f{or last three months:
Sales for month 1: 1200
Sales for month 2: 500
Sales for month 3: 6000

Hardware Item Details:
Item Title: Laptop
Item Manufacturer: Irfa
Item price: 30000Rs
Sales f{or last three months:
Sales for month 1: 2200
Sales for month 2: 100
Sales for month 3: 12000

Software Item Details:
Item Title: Windows
Item price: 95000Rs
Item Operating System: Microsoft
Sales for last three months:
Sales for month 1: 9200
Sales for month 2: 5000
Sales for month 3: 16000
```

**Conclusion:**

In this lab, we were able to understand the concepts of dynamic binding i.e.; polymorphism and use it in our coding to make more efficient codes. Other than that, we were also able to understand and use pure virtual functions.