OBJECT ORIENTED PROGRAMMING

MC307 Lab

ANEESH PANCHAL

2K20/MC/21

Department of Applied Mathematics,

Delhi Technological University

Submitted To –

Prof. Aditya Kaushik

and Ms. Shivani Jain

INDEX

S. No.	Experiment	Date	Sign & Remark
01.	Create a class named “Time” in C++ with three data members namely hour, minute and seconds. Create two member functions of the class namely “setTime” that sets the hour, minute and seconds for any object; and “print” function that prints the hour, minute, and seconds value for the object. Test the class by creating its two objects in main.	11/08/23	
02.	Write a program in C++ demonstrating the use of constructor (with no arguments), constructor (with parameterized arguments) and destructor. Create objects of that class in main and test those objects with these constructors and destructor function.	18/08/23	
03.	Write a C++ program illustrating the operator overloading of binary operators (addition and subtraction), and unary operators (prefix and postfix increment).	25/08/23	
04.	Implement inheritance in C++ program. Create a base class named Person (with data members first_name, and last_name). Derive a class named Employee from base class Person. The Employee class should include a data member named employee_id. Further a class named Manager should be derived from the class Employee and the Manager class should include two data members named employee_designation and employee_salary. Create at least three objects of class Manager, specify the values of all their attributes, and determine which manager has the highest salary.	01/09/23	
05.	Write a program in C++ demonstrating class templates.	08/09/23	
06.	Write a program in C++ demonstrating rethrowing an exception.	15/09/23	

S. No.	Experiment	Date	Sign & Remark
07.	Create an inheritance hierarchy (any inheritance example of your choice) in JAVA. Create one super class and three sub classes should inherit from that super class. Also highlight the concept of function overriding and function overloading in your program.	15/09/23	
08.	Write a JAVA program explaining the working of constructors along with the super keyword.	22/09/23	
09.	Write a Java program consisting of abstract classes and abstract method(s). The first concrete sub class in the inheritance hierarchy must implement the abstract method(s). Create the objects of the concrete class and test your program.	06/10/23	
10.	Write a Java program explaining the working of static methods in a class.	13/10/23	

Experiment 1

Aim:

Create a class named “Time” in C++ with three data members namely hour, minute and seconds. Create two member functions of the class namely “setTime” that sets the hour, minute and seconds for any object; and “print” function that prints the hour, minute, and seconds value for the object. Test the class by creating its two objects in main.

Code:

```
#include<iostream>
class Time{
    int hour, minute, seconds;
    public:
        Time(): hour(0),minute(0),seconds(0){}
        void setTime(int hr, int min, int sec){
            hour = hr;
            minute = min;
            seconds = sec;
        }
        void print() const{
            if(hour<0 || hour>24 || minute<0 || minute>60 || seconds<0 || seconds>60)
                std::cout<<"Invalid Input"<<std::endl;
            else
                std::cout<<hour<<":"<<minute<<":"<<seconds<<std::endl;
        }
};

int main(){
    Time t1, t2;
    t1.setTime(12,30,45);
    t2.setTime(5,15,5);
    std::cout<<"Time of object t1: ";
    t1.print();
    std::cout<<"Time of object t2: ";
    t2.print();
    return 0;
}
```

Output:

```
PS E:\Codes\MC307 OOPS> cd "e:\Codes\MC307 OOPS\" ; if ($?) { g++ Exp_1.cpp -o Exp_1 } ; if ($?) { .\Exp_1 }

Time of object t1: 12:30:45
Time of object t2: 5:15:5

PS E:\Codes\MC307 OOPS>
```

Experiment 2

Aim:

Write a program in C++ demonstrating the use of constructor (with no arguments), constructor (with parameterized arguments) and destructor. Create objects of that class in main and test those objects with these constructors and destructor function.

Code:

```
#include<iostream>
class Time{
    int hour, minute, seconds;
    public:
        Time(): hour(0),minute(0),seconds(0){
            std::cout<<"Default constructor called !!!"<<std::endl;
        }
        Time(int h, int m, int s): hour(h),minute(m),seconds(s){
            std::cout<<"Parameterized constructor called !!!"<<std::endl;
        }
        ~Time(){
            std::cout<<"Destructor called for Time: ";
            print();
        }
        void print() const{
            if(hour<0 || hour>24 || minute<0 || minute>60 || seconds<0 || seconds>60)
                std::cout<<"Invalid Input"<<std::endl;
            else
                std::cout<<hour<<":"<<minute<<":"<<seconds<<std::endl;
        }
};

int main(){
    Time t1;
    t1.print();
    Time t2(12, 15, 30);
    t2.print();
    return 0;
}
```

Output:

```
PS E:\Codes\MC307 OOPS> cd "e:\Codes\MC307 OOPS\" ; if ($?) { g++ Exp_2.cpp -o Exp_2 } ; if ($?) { .\Exp_2 }

Default constructor called !!!
0:0:0

Parameterized constructor called !!!
12:15:30

Destructor called for Time: 12:15:30

Destructor called for Time: 0:0:0

PS E:\Codes\MC307 OOPS>
```

Experiment 3

Aim:

Write a C++ program illustrating the operator overloading of binary operators (addition and subtraction), and unary operators (prefix and postfix increment).

Code:

```
#include<iostream>
class operatorOverloading{
    int value;
    public:
        operatorOverloading(int v=0): value(v){}

        operatorOverloading operator +(const operatorOverloading &c){
            return operatorOverloading(value + c.value);
        }
        operatorOverloading operator -(const operatorOverloading &c) {
            return operatorOverloading(value - c.value);
        }
        operatorOverloading &operator ++(){
            value++;
            return *this;
        }
        operatorOverloading operator ++(int){
            operatorOverloading temp = *this;
            value++;
            return temp;
        }
        void display() const{
            std::cout<<value;
        }
};

int main() {
    operatorOverloading c1(5), c2(3), c3;

    std::cout<<"Initial values: ";
    std::cout<<"c1 = ";
    c1.display();
    std::cout<<", c2 = ";
    c2.display();
    std::cout<<", c3 = ";
    c3.display();

    // Binary + operator
    c3 = c1 + c2;
    std::cout<<"After addition: ";
    c3.display();

    // Binary - operator
    c3 = c1 - c2;
```

```
    std::cout<<"After subtraction: ";
    c3.display();

    // Prefix ++ operator
    ++c1;
    std::cout<<"After prefix increment on c1: ";
    c1.display();

    // Postfix ++ operator
    c2++;
    std::cout<<"After postfix increment on c2: ";
    c2.display();
    return 0;
}
```

Output:

```
PS E:\Codes\MC307 OOPS> cd "e:\Codes\MC307 OOPS\" ; if ($?) { g++ Exp_3.cpp -o Exp_3 } ; if ($?) { .\Exp_3 }

Initial values: c1 = 5, c2 = 3, c3 = 0

After addition: 8

After subtraction: 2

After prefix increment on c1: 6

After postfix increment on c2: 4

PS E:\Codes\MC307 OOPS>
```

Experiment 4

Aim:

Implement inheritance in C++ program. Create a base class named Person (with data members first_name, and last_name). Derive a class named Employee from base class Person. The Employee class should include a data member named employee_id. Further a class named Manager should be derived from the class Employee and the Manager class should include two data members named employee_designation and employee_salary. Create at least three objects of class Manager, specify the values of all their attributes, and determine which manager has the highest salary.

Code:

```
#include<iostream>
#include<string>
class Person{
    protected:
        std::string first_name;
        std::string last_name;
    public:
        Person(const std::string &fname, const std::string &lname):
            first_name(fname), last_name(lname){}
        void displayPerson() const{
            std::cout<<"Name: "<<first_name<<" "<<last_name<<std::endl;
        }
};

class Employee: public Person{
    protected:
        int employee_id;
    public:
        Employee(const std::string &fname, const std::string &lname, int id):
            Person(fname, lname), employee_id(id){}
        void displayEmployee() const{
            displayPerson();
            std::cout<<"Employee ID: "<<employee_id<<std::endl;
        }
};

class Manager: public Employee{
    private:
        std::string employee_designation;
        double employee_salary;
    public:
        Manager(const std::string &fname, const std::string &lname, int id, const
std::string &designation, double salary):
            Employee(fname, lname, id), employee_designation(designation),
employee_salary(salary){}
        void displayManager() const{
            displayEmployee();

            std::cout<<"Designation: "<<employee_designation<<std::endl;
```

```
            std::cout<<"Salary: $"<<employee_salary<<std::endl;
        }
        double getSalary() const{
            return employee_salary;
        }
};

int main(){
    Manager m1("Dhruv", "Rana", 1001, "Tech Head", 80000);
    Manager m2("Aneesh", "Panchal", 1002, "Research Head", 85000);
    Manager m3("Anshul", "Aggarwal", 1003, "Corporate Head", 80000);

    m1.displayManager();
    m2.displayManager();
    m3.displayManager();

    if(m1.getSalary()>m2.getSalary() && m1.getSalary()>m3.getSalary()){
        std::cout<<"Highest salary: $"<<m1.getSalary()<<std::endl;
        std::cout<<"Manager with highest salary have ";
        m1.displayPerson();
    }
    else if(m2.getSalary()>m1.getSalary() && m2.getSalary()>m3.getSalary()){
        std::cout<<"Highest salary: $"<<m2.getSalary()<<std::endl;
        std::cout << "Manager with highest salary have ";
        m2.displayPerson();
    }
    else{
        std::cout<<"Highest salary: $"<<m3.getSalary()<<std::endl;
        std::cout<<"Manager with highest salary have ";
        m3.displayPerson();
    }
    return 0;
}
```

Output:

```
PS E:\Codes\MC307 OOPS> cd "e:\Codes\MC307 OOPS\" ; if ($?) { g++ Exp_4.cpp -o Exp_4 } ; if ($?) { .\Exp_4 }

Name: Dhruv Rana
Employee ID: 1001
Designation: Tech Head
Salary: $75000

Name: Aneesh Panchal
Employee ID: 1002
Designation: Research Head
Salary: $85000

Name: Anshul Aggarwal
Employee ID: 1003
Designation: Corporate Head
Salary: $80000

Highest salary: $85000
Manager with highest salary have Name: Aneesh Panchal

PS E:\Codes\MC307 OOPS>
```

Experiment 5

Aim:

Write a program in C++ demonstrating class templates.

Code:

```cpp
#include<iostream>
template<typename Template_example>
class Class_example{
    private:
        Template_example item;
    public:
        Class_example(Template_example i): item(i){}
        Template_example getItem() const{
            return item;
        }
        void setItem(Template_example i){
            item = i;
        }
        void display() const{
            std::cout<<"Item in the Specialized Class: "<<item<<std::endl;
        }
};

int main(){
    Class_example<int> int_example(5);
    std::cout<<"Integer Example: ";
    int_example.display();

    Class_example<double> double_example(3.14);
    std::cout<<"Double Example: ";
    double_example.display();

    Class_example<std::string> string_example("Hello, Templates !");
    std::cout<<"String Example: ";
    string_example.display();
    return 0;
}
```

Output:

```
PS E:\Codes\MC307 OOPS> cd "e:\Codes\MC307 OOPS\" ; if ($?) { g++ Exp_5.cpp -o Exp_5 } ; if ($?) { .\Exp_5 }

Integer Example: Item in the Specialized Class: 5

Double Example: Item in the Specialized Class: 3.14

String Example: Item in the Specialized Class: Hello, Templates !

PS E:\Codes\MC307 OOPS>
```

Experiment 6

Aim:

Write a program in C++ demonstrating rethrowing an exception.

Code:

```
#include<iostream>
void divide(int num, int den){
    try{
        if(den == 0){
            throw std::runtime_error("Denominator is zero !");
        }
        double result = static_cast<double>(num)/den;
        std::cout<<"Result: "<<result<<std::endl<<std::endl;
    }
    catch(const std::runtime_error &err){
        std::cerr<<"Exception caught in divide(): "<<err.what()<<std::endl<<std::endl;
        throw;
    }
}

int main(){
    try{
        divide(10, 2);
        divide(10, 0);
    }
    catch(const std::exception& err){
        std::cerr<<"Exception caught in main(): "<<err.what()<<std::endl<<std::endl;
    }
    return 0;
}
```

Output:

```
PS E:\Codes\MC307 OOPS> cd "e:\Codes\MC307 OOPS\" ; if ($?) { g++ Exp_6.cpp -o Exp_6 } ; if ($?) { .\Exp_6 }

Result: 5

Exception caught in divide(): Denominator is zero !

Exception caught in main(): Denominator is zero !


PS E:\Codes\MC307 OOPS>
```

Experiment 7

Aim:

Create an inheritance hierarchy (any inheritance example of your choice) in JAVA. Create one super class and three sub classes should inherit from that super class. Also highlight the concept of function overriding and function overloading in your program.

Code:

```
class Animal{
    String name;
    Animal(String name){
        this.name = name;
    }

    void sound(){
        System.out.println("General Animal Sound");
    }
    void sound(String Sound_String){
        System.out.println(Sound_String);
    }
    void habitat(){
        System.out.println(name + ":");
        System.out.println("Multiple Habitats");
    }
}

class Mammal extends Animal{
    Mammal(String name){
        super(name);
    }

    @Override
    void habitat(){
        System.out.println(name + ":");
        System.out.println("Mammals lives on land");
    }
}

class Bird extends Animal{
    Bird(String name){
        super(name);
    }

    @Override
    void habitat(){
        System.out.println(name + ":");
        System.out.println("Birds fly in air");
    }
}
```

```java
class Fish extends Animal{
    Fish(String name){
        super(name);
    }

    @Override
    void habitat(){
        System.out.println(name + ":");
        System.out.println("Fishes lives underwater");
    }
}

public class Exp_7{
    public static void main(String[] args){
        Animal animal = new Animal("Animal");
        Mammal lion = new Mammal("Lion");
        Bird eagle = new Bird("Eagle");
        Fish salmon = new Fish("Salmon");
        animal.habitat();

        System.out.println("without parameter:");
        lion.sound();
        System.out.println("with parameter:");
        lion.sound("Roar!");

        lion.habitat();
        eagle.habitat();
        salmon.habitat();
    }
}
```

Output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWin
dows

PS E:\Codes\MC307 OOPS>  & 'C:\Users\ANEESH PANCHAL\AppData\Local\Programs\Eclipse Ad
optium\jdk-17.0.8.101-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages'
 '-cp' 'C:\Users\ANEESH PANCHAL\AppData\Roaming\Code\User\workspaceStorage\0cd545e72b
3a12038184401722514136\redhat.java\jdt_ws\MC307 OOPS_2a619e42\bin' 'Exp_7'

Animal:
Multiple Habitats

without parameter:
General Animal Sound
with parameter:
Roar!

Lion:
Mammals lives on land

Eagle:
Birds fly in air

Salmon:
Fishes lives underwater

PS E:\Codes\MC307 OOPS>
```

Experiment 8

Aim:

Write a JAVA program explaining the working of constructors along with the super keyword.

Code:

```
class Vehicle {
    String brand;
    Vehicle(String brand) {
        this.brand = brand;
        System.out.println("Superclass Constructor: Vehicle of brand " + brand + " is
created.");
    }
    void displayBrand() {
        System.out.println("Vehicle brand: " + brand);
    }
}

class Car extends Vehicle {
    String carType;
    Car(String brand, String carType) {
        super(brand);
        this.carType = carType;
        System.out.println("Subclass Constructor: It's a " + carType + ".");
    }
    void displayDetails() {
        System.out.println("It's a " + brand + " " + carType + ".");
    }
}

public class Exp_8 {
    public static void main(String[] args) {
        Car sedanCar = new Car("Toyota", "Sedan");
        sedanCar.displayBrand();
        sedanCar.displayDetails();
    }
}
```

Output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Codes\MC307 OOPS>  & 'C:\Users\ANEESH PANCHAL\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.8.101-hots
pot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ANEESH PANCHAL\AppData\Roaming\Code
\User\workspaceStorage\0cd545e72b3a12038184401722514136\redhat.java\jdt_ws\MC307 OOPS_2a619e42\bin' 'Exp_8'

Superclass Constructor: Vehicle of brand Toyota is created.
Subclass Constructor: It's a Sedan.

Vehicle brand: Toyota
It's a Toyota Sedan.

PS E:\Codes\MC307 OOPS>
```

Experiment 9

Aim:

Write a Java program consisting of abstract classes and abstract method(s). The first concrete sub class in the inheritance hierarchy must implement the abstract method(s). Create the objects of the concrete class and test your program.

Code:

```
abstract class Shape {
    String color;
    Shape(String color) {
        this.color = color;
    }
    abstract double area();
    void displayColor() {
        System.out.println("Color: " + color);
    }
}

class Circle extends Shape {
    double radius;
    Circle(String color, double radius) {
        super(color);
        this.radius = radius;
    }
    @Override
    double area() {
        return Math.PI * radius * radius;
    }
}

public class Exp_9 {
    public static void main(String[] args) {
        Circle circle = new Circle("Red", 5);
        circle.displayColor();
        System.out.println("Area of circle: " + circle.area());
    }
}
```

Output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Codes\MC307 OOPS>  & 'C:\Users\ANEESH PANCHAL\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.8.101-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ANEESH PANCHAL\AppData\Roaming\Code\User\workspaceStorage\0cd545e72b3a12038184401722514136\redhat.java\jdt_ws\MC307 OOPS_2a619e42\bin' 'Exp_9'

Color: Red
Area of circle: 78.53981633974483

PS E:\Codes\MC307 OOPS>
```

Experiment 10

Aim:

Write a Java program explaining the working of static methods in a class.

Code:

```
class Counter{
    static int totalCount = 0;
    int objectCount = 0;
    Counter(){
        totalCount++;
        objectCount++;
    }

    static void displayTotalCount(){
        System.out.println("Total objects created: " + totalCount);
    }
    void displayObjectCount(){
        System.out.println("Object count: " + objectCount);
    }
}

public class Exp_10{
    public static void main(String[] args){
        Counter obj1 = new Counter();
        obj1.displayObjectCount();
        Counter.displayTotalCount();
        Counter obj2 = new Counter();
        obj2.displayObjectCount();
        Counter.displayTotalCount();
        obj1.totalCount = 5;
        System.out.println("Total count from obj1: " + obj1.totalCount);
        System.out.println("Total count from obj2: " + obj2.totalCount);
    }
}
```

Output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Codes\MC307 OOPS>  & 'C:\Users\ANEESH PANCHAL\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.8.101-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ANEESH PANCHAL\AppData\Roaming\Code\User\workspaceStorage\0cd545e72b3a12038184401722514136\redhat.java\jdt_ws\MC307 OOPS_2a619e42\bin' 'Exp_10'

Object count: 1

Total objects created: 1

Object count: 1

Total objects created: 2

Total count from obj1: 5
Total count from obj2: 5

PS E:\Codes\MC307 OOPS>
```