National University
of computer and emerging sciences

# Lab 6

| Items | Description |
|---|---|
| Course Title | Object Oriented Programming |
| Lab Title | CLasses |
| Duration | 3 Hours |
| Tools | Eclipse/ C++ |
| Objective | To get familiar with the use of different concepts within CLasses in c++ |

## 1. Parameterized Constructor:

A parameterized constructor is a constructor in a class that takes parameters, allowing objects to be initialized with specific values at the time of creation.

```
#include <iostream>

using namespace std;

class Point {

private:

    int x, y;



public:
```

```
    // Parameterized constructor
    Point(int a, int b) {
        x = a;
        y = b;
    }
    void display() {
        cout << "x: " << x << " y: " << y << endl;
    }
};
int main() {
    Point p(10, 20); // Object initialized using parameterized constructor
    Point p; // Error as the only constructor available requires 2 parameters
    p.display();
    return 0;
}
```

# 2. Constant Member Function:

A constant member function ensures that it doesn't modify any class members. It is declared by adding `const` after the function signature.

```
#include <iostream>

using namespace std;

class Point {

private:

    int x, y;

public:

    Point(int a, int b) {

        x = a;

        y = b;

    }

    // Constant member function

    void display() const {

       cout << "x: " << x << " y: " << y << endl;

    }

};

int main() {
```

```
    Point p(10, 20);

    p.display(); // Call constant member function

    return 0;

}
```

# 3. Objects with Pointers:

You can have pointers inside a class that point to dynamically allocated memory. Care must be taken to properly manage the memory, especially in the constructor and destructor.

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

class Point {
private:
    int *x, *y;

public:
    Point(int a, int b) {

        x = a;

        y = b;

    }


    ~Point() {
        delete x;
```

```
        delete y;
    }

    void display() const {
        cout << "x: " << *x << " y: " << *y << endl;
    }
};

int main() {
    Point p(10, 20);
    p.display();
    return 0;
}
```

# 4. Private Member Function:

A private member function is only accessible within the class and cannot be called from outside the class directly. It is typically used to support other public functions.

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

class Point {
private:
    int x, y;

    // Private member function
    void showPrivate() const {
```

```
        cout << "Private function: x = " << x << " y = "
<< y << endl;
    }

public:
    Point(int a, int b) {

        x = a;

        y = b;

    }


    void display() const {
        showPrivate();  // Private function called from
public function
    }
};

int main() {
    Point p(10, 20);
    p.display();  // Indirectly calling private function
    return 0;
}
```

# 5. Inline and Out of Line Function:

An inline function is defined inside the class and is expanded at the point of the call, whereas an out-of-line function is defined outside the class using the scope resolution operator `::`.


```cpp
#include <iostream>

using namespace std;


class Point {

private:

    int x, y;


public:

    Point(int a, int b) {

        x = a;

        y = b;

    }


    // Inline function

    void display() const {

        cout << "Inline: x = " << x << " y = " << y <<
endl;

    }
```

```
    // Out-of-line function declaration
    void move(int, int);
};

// Out-of-line function definition
void Point::move(int a, int b) {
    x = a;
    y = b;
    cout << "Out of line: Moved to x = " << x << " y = "
<< y << endl;
}

int main() {
    Point p(10, 20);
    p.display();
    p.move(30, 40);
    p.display();
    return 0;
}
```

National University
of computer and emerging sciences

# Tasks

## 1. Task: Bank Account Class with Parameterized Constructor

- Create a class `BankAccount` with attributes like `accountNumber`, `accountHolderName`, and `balance`.
- Implement a **parameterized constructor** to initialize the account details.
- Include functions to:
    - Deposit money
    - Withdraw money
    - Display the account details.

## 2. Task: Employee Class with Private Functions

- Create a class `Employee` with private data members for `name`, `salary`, and `age`.
- Add a **private member function** `calculateBonus` that calculates a bonus based on the employee's salary i:e salary * 0.25.
- Provide public functions to:
    - Set the employee's details.
    - Call the private function and return the bonus amount.
    - Display employee details.

## 3. Task: DynamicArray Class with Pointers

- Create a class `DynamicArray` that manages a dynamically allocated integer array using pointers.
- Implement:
    - A **parameterized constructor** to initialize the array with a given size.
    - Methods to:
        - **Add** elements at the end of the array (resize the array dynamically).
        - **Remove** elements from the end of the array.
        - **Insert** an element at a specific index.
        - **Delete** an element from a specific index.
        - **Resize** the array when the capacity is exceeded.
    - A **destructor** to free the dynamically allocated memory.
    - A **constant member function** to display the array contents.