C++ Concepts with Examples

Inheritance

Inheritance allows one class to acquire the properties of another.

Example:

```
// Example: Rectangle inheriting Shape
#include <iostream>
using namespace std;

class Shape {
public:
    void displayShape() {
        cout << "This is a Shape" << endl;
    }
};

class Rectangle : public Shape {   // Rectangle inherits
Shape
public:
    void displayRectangle() {
        cout << "This is a Rectangle" << endl;
    }
};

int main() {
    Rectangle r;
    r.displayShape();      // from base class
    r.displayRectangle();  // from derived class
    return 0;
}
```

Public, Private, Protected Inheritance

Access modifiers control how members are inherited.

Example:

```
// Example with Circle and access specifiers
#include <iostream>
using namespace std;

class Shape {
public:
    int sides = 0;
protected:
    string color = "Red";
private:
    string secret = "Hidden";
};

class Circle : public Shape {   // public inheritance
public:
    void show() {
        cout << "Sides: " << sides << endl;   // accessible
        cout << "Color: " << color << endl;   // accessible
(protected)
```

```
        cout << secret;  // not accessible
    }
};

int main() {
    Circle c;
    c.show();
    return 0;
}
```

Forms of Inheritance

Different types of inheritance can be demonstrated with shapes.

Example:

```
// Example: Single, Multilevel, Hierarchical
#include <iostream>
using namespace std;

class Shape {
public:
    void showShape() { cout << "This is a Shape" << endl; }
};

class Rectangle : public Shape {
public:
    void showRectangle() { cout << "This is a Rectangle" <<
endl; }
};

class Square : public Rectangle {   // Multilevel
public:
    void showSquare() { cout << "This is a Square" << endl; }
};

class Circle : public Shape {       // Hierarchical
public:
    void showCircle() { cout << "This is a Circle" << endl; }
};

int main() {
    Square sq;
    sq.showShape();     // from Shape
    sq.showRectangle(); // from Rectangle
    sq.showSquare();    // from Square

    Circle c;
    c.showShape();      // from Shape
    c.showCircle();     // from Circle
    return 0;
}
```

Hierarchical Inheritance

One base class inherited by multiple derived classes.

Example:

```
// Example: Shape -> Rectangle, Circle
#include <iostream>
using namespace std;

class Shape {
public:
    void display() {
        cout << "This is a Shape" << endl;
    }
};

class Rectangle : public Shape {
public:
    void displayRectangle() {
        cout << "Rectangle is a Shape" << endl;
    }
};

class Circle : public Shape {
public:
    void displayCircle() {
        cout << "Circle is a Shape" << endl;
    }
};
```

```
int main() {
    Rectangle r;
    Circle c;
    r.display();        // from Shape
    r.displayRectangle();
    c.display();        // from Shape
    c.displayCircle();
    return 0;
}
```

Multilevel Inheritance

A derived class further acts as a base class for another class.

Example:

```
// Example: Shape -> Rectangle -> Square
#include <iostream>
using namespace std;

class Shape {
public:
    void displayShape() {
        cout << "This is a Shape" << endl;
    }
};

class Rectangle : public Shape {
public:
    void displayRectangle() {
        cout << "This is a Rectangle" << endl;
    }
};

class Square : public Rectangle {
public:
    void displaySquare() {
        cout << "This is a Square" << endl;
    }
};

int main() {
    Square s;
    s.displayShape();      // from Shape
    s.displayRectangle();  // from Rectangle
    s.displaySquare();     // from Square
    return 0;
}
```