CL23 10 22　インスタンス管理

目次

インスタンスの管理

C#やJavaにはガベージコレクションが存在する。

C++特有のオブジェクト指向の考え方において、

1つの課題となりえるのはインスタンス管理である

newはあるけど、自動的にdeleteをしてくれる仕組み

スマートポインタ

C++11スマートポインタ入門 - Qiita

本解説では、スマートポインタについて初めて学ぶ人を対象に、C++11で追加された3種のスマートポインタの機能と使い方、および3種をどのように考えて使うかについて、初歩的な解

https://qiita.com/hmito/items/db3b14917120b285112f

オブジェクトへのアクセス

問

ここに猫クラスがある。せっかくなのでこの猫に名前をつけられるようにしようと思う。

そこで。。。

```
class Cat{
    std::string m_name;
public;
    void SetName(string s){m_name = s;}
```

```
    void Reference(string s){&m_name = s;}
    Cat() {
        Cat = "name";
        }
    Cat() : m_name("name"){}
    void InputName(){cin>>m_name;}
};
```

1. メンバ変数をpublicにして直接代入

```
{
public:
    string m_name ;
}

int main(){
    Cat cat;
    cat.m_name = "kuro";
}
```

解説

この方法はやめてほしいやり方。

インターフェイスという考えが重要

インターフェイスとはそのオブジェクトができることのことで、

基本的にprivateであったり、そもそもメソッドが存在しない場合は開示されていないということになる。

2.コンストラクタでの設定

```
{
    string m_name;
public:
    Cat() : m_name("ねこちゃん"){}
}
```

解説

オブジェクトはインスタンスを持った時点で、
その動作が保証されているべきです。

どのような順番でメソッドを実行下としても、アクセス違反のような
テクニカルなエラーになってしまってはいけません。

よってコンストラクタで内部で管理しているデータなどを動作が保証されるレベルまで
初期化をする必要があります。

2.5引数付きコンストラクタでの設定

```
class Cat
{
    string m_name;
public:
    Cat(const string name){m_name = name;}
}
```

解説

引数付きコンストラクタでの設定
「コンストラクタでの設定」での解説に加え、
どのようにインスタンスを設定して生成したいかを指定できるのが特徴です。

さらにデフォルトコンストラクタを封印（privateアクセス制限したり、= deleteにより暗黙定義を禁止したり）することにより、必ず設定した上でインスタンスを生成してほしいという一種の強制力をもたせることが可能です。

3.アクセス用メソッド（セッター）での設定

```
class Cat{
    string m_name;
public:
    void SetName(string &name){m_name = name;}
};

int main(){
    Cat cat;
    cat.SetName("くま");
}
```

解説

- 一番スタンダードなメンバ変数への設定方法。
- 何度でも設定できる。
- 逆にセッターが用意されていないものは、そのメンバ変数への直接アクセスはできない。

番外編　～コピーコンストラクタ～

```
int main(){
    Cat cat_a("たま");
    Cat cat_b(cat_a);
    Cat cat_c = cat_a;
    cat_b = cat_a;
}
```

解説

コピーやコピーコンストラクタによって

元になるオブジェクトと同じ値が設定されていた状態でインスタンスを生成したり、

値を上書きしたりすることができます。

ほかのオブジェクト指向言語では、

cloneメソッドやcopyメソッドのように自分自身を複製するメソッドを必ずもつ仕様の言語もあります。

noncopyable

こいつを継承するとコピーできなくなる。

C++標準での機能には無い。

なので、自分で自作する必要がある。

オブジェクトへのアクセス　～まとめ～

このように１つのメンバ変数へ値を設定する方法はいくつも存在しており、その方法１つ１つに特徴があります。

自分が作りたいオブジェクトのメンバ変数がどのようなものか判断し、適切な方法で設定できるよう選択する必要があります。

また、これらの方法を併用することも当然考えることができます。