静的メンバ

- 静的メンバとインスタンスメンバ

  これまではインスタンス生成を行うことで
  ・メンバ変数
  ・メンバ関数
  を利用してきた

  こういったメンバ変数/関数をインスタンスメンバ
  と呼ぶ

静的メンバ

- 静的メンバとインスタンスメンバ

静的メンバはインスタンス生成を行わずに
・メンバ変数
・メンバ関数
を利用できる仕組み

ではどう使うのか？

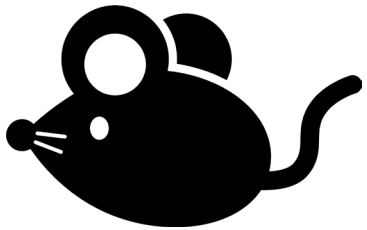
静的メンバ

- 教科書P152~153 **Sample405**

- C++作業フォルダ内に**Sample405**フォルダを作成
```
mkdir Sample405
cd Sample405
```

- rat.h, rat.cpp, main.cppを作成
```
copy nul rat.h
copy nul rat.cpp
copy nul main.cpp
```

静的メンバ



- rat.h（Sample405）

```
#pragma once

class Rat {
public:
	Rat();
	~Rat();
	static void showNum(); //静的メンバ関数
	void squeak();
private:
	int m_id;
	static int s_count;	//静的メンバ変数
};
```

静的メンバ

- rat.cpp（Sample405）

```
#include “rat.h”
#include <iostream>
using namespace std;
int Rat::s_count = 0;

Rat::Rat() : m_id(0) {
  s_count++;
  m_id = s_count;
}
Rat::~Rat() {
  cout << “ネズミ:” << m_id
    << “消去” << endl;
  s_count--;
}
```

```
void Rat::showNum() {
  cout << “現在のネズミの数は,”
    << s_count << “匹です。”
    << endl;
}
void Rat::squeak() {
  cout << m_id << “:” <<
    “チューチュー” << endl;
}
```

静的メンバ

- main.cpp（Sample405）

```
#include “rat.h”

int main()
{
  Rat* r1, * r2, * r3;
  r1 = new Rat();
  r1->squeak();
  Rat::showNum();
  r2 = new Rat();
  r3 = new Rat();
  r2->squeak();
  r3->squeak();
  Rat::showNum();
  delete r1;
  delete r2;
  Rat::showNum();
  delete r3;
  Rat::showNum();
  return 0;
}
```

静的メンバ

- コンパイルの仕方

  コマンドプロンプトで次のコマンドを入力する

```
cl /EHsc main.cpp rat.cpp
```

  成功したら、main.exeを実行して結果を確認

静的メンバ

- rat.h（Sample405）

```
#pragma once

class Rat {
public:
	Rat();
	~Rat();
	static void showNum(); //静的メンバ関数
	void squeak();
private:
	int m_id;
	static int s_count;	//静的メンバ変数
};
```

静的メンバは
static
を付けて宣言する

静的メンバ変数はクラス定義時に初期化不可

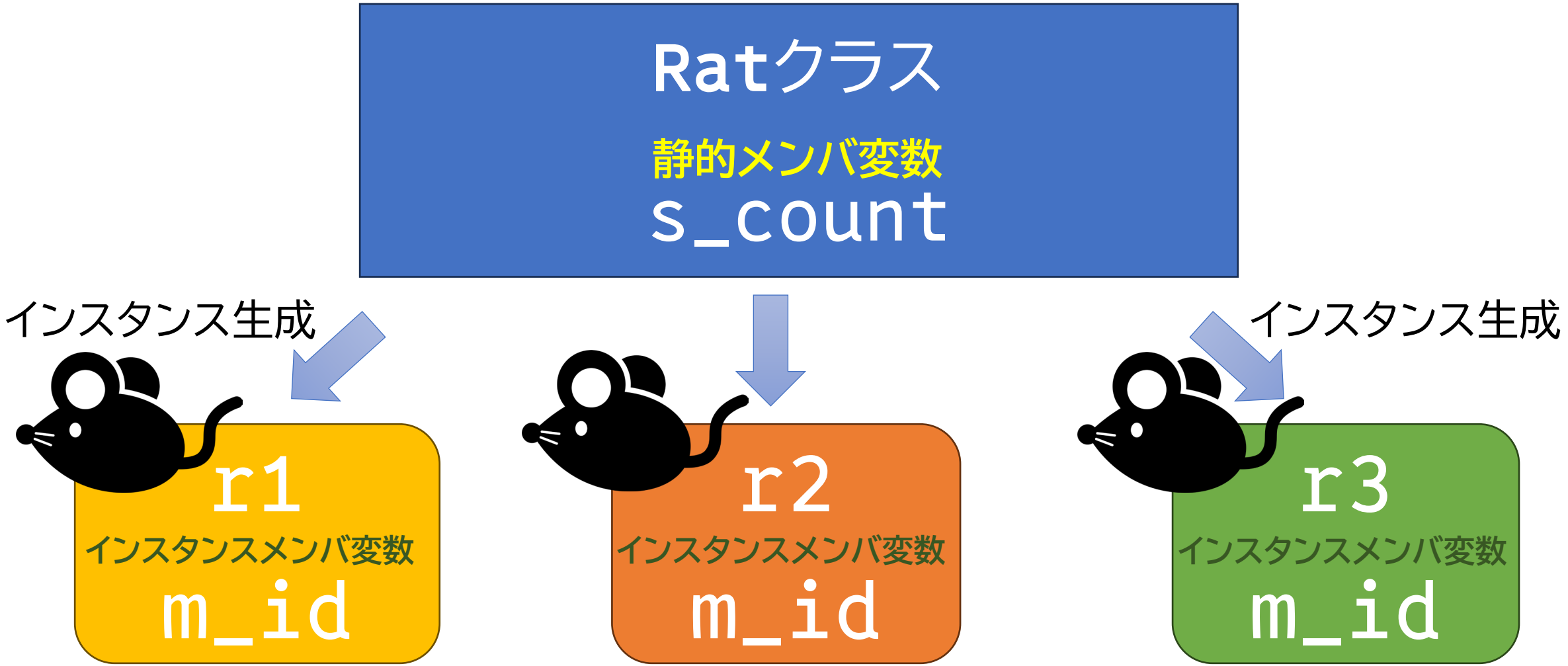
静的メンバ

- 静的メンバはインスタンス生成を行わずに
  ・メンバ変数
  ・メンバ関数
  を利用できる

  クラス自体が所有している変数や関数なので、
  インスタンス生成前や生成後でも
  変数を参照したり、関数を実行できる！

静的メンバ

静的メンバ変数はクラス自体が
持っている変数であって参照可能だが
インスタンスのメンバではない



静的メンバ

- rat.cpp（Sample405）

静的メンバの初期化は
クラス定義外でのみ可能

```
int Rat::s_count = 0;
Rat::Rat() : m_id(0) {
  s_count++;
  m_id = s_count;
}
Rat::~Rat() {
  cout << “ネズミ:” << m_id
    << “消去” << endl;
  s_count--;
}
```

```
void Rat::showNum() {
  cout << “現在のネズミの数は,”
    << s_count << “匹です。”
    << endl;
}
void Rat::squeak() {
  cout << m_id << “:” <<
    “チューチュー” << endl;
}
```

静的メンバ

- rat.cpp（Sample405）

```
#include “rat.h”
#include <iostream>
using namespace std;

int Rat::s_count = 0;
Rat::Rat() : m_id(0) {
  s_count++;
  m_id = s_count;
}
Rat::~Rat() {
  cout << “ネズミ:”
    << “消去” << endl;
  s_count--;
}
```

```
void Rat::showNum() {
  cout << “現在のネズミの数は,”
    << s_count << “匹です。”
    << endl;
}
void Rat::squeak() {
  cout << m_id << “:” <<
                    ndl;
```

s_countはprivateのため、
クラス内の関数から参照可能

静的メンバ

コンストラクタ
s_countをインスタンス生成時に＋1する
そのときのs_count値を自分のID番号(m_id)にする

```
int Rat::s_count = 0;
Rat::Rat() : m_id(0) {
  s_count++;
  m_id = s_count;
}
Rat::~Rat() {
  cout << “ネズミ:” << m_id
    << “消去” << endl;
  s_count--;
}
```

```
howNum() {
現在のネズミの数は,”
unt << “匹です。”
    << endl;
}
void Rat::squeak() {
  cout << m_id << “:” <<
    “チューチュー” << endl;
}
```

静的メンバ

- rat.cpp（Sample405）

```
#include “rat.h”
#include <iostream>
using namespace std;

int Rat::s_count = 0;
Rat::Rat() : m_id(0) {
}
Rat::~Rat() {
   cout << “ネズミ:” << m_id
      << “消去” << endl;
   s_count--;
}
```

```
void Rat::showNum() {
   cout << “現在のネズミの数は,”
      << s_count << “匹です。”
      << endl;
}

squeak() {
m_id << “:” <<
   “チューチュー” << endl;
}
```

デストラクタ

s_countをインスタンス消去時に－1する

静的メンバ

- rat.cpp (

静的メンバ関数showNum()
インスタンスを生成せずに呼び出し可能

```
#include “rat.h”
#include <iostream>
using namespace std;

int Rat::s_count = 0;
Rat::Rat() : m_id(0) {
  s_count++;
  m_id = s_count;
}
Rat::~Rat() {
  cout << “ネズミ:” << m_id
    << “消去” << endl;
  s_count--;
}
```

```
void Rat::showNum() {
  cout << “現在のネズミの数は,”
    << s_count << “匹です。”
    << endl;
}
void Rat::squeak() {
  cout << m_id << “:” <<
    “チューチュー” << endl;
}
```

静的メンバ

- rat.cpp

```
#include “rat.h”
#include <iostream>
using namespace std;

int Rat::s_cou
Rat::Rat() : m
  s_count++;
  m_id = s_count;
}
Rat::~Rat() {
  cout << “ネズミ:” << m_id
    << “消去” << endl;
  s_count--;
}
```

```
void Rat::showNum() {
  cout << “現在のネズミの数は,”
```

メンバ関数squeak()
インスタンスを生成しないと呼び出し不可！

```
void Rat::squeak() {
  cout << m_id << “:” <<
    “チューチュー” << endl;
}
```

静的メンバ

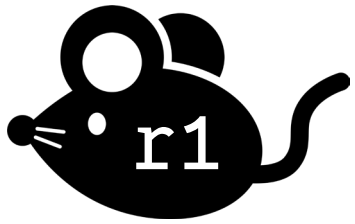
- main.cpp（Sample405）

```
#i
in
{
  Rat* r1, * r2, * r3;
  r1 = new Rat();
  r1->squeak();
  Rat::showNum();
  r2 = new Rat();
  r3 = new Rat();
  r2->squeak();
  r3->squeak();

  Rat::showNum();
  delete r1;
  delete r2;
  Rat::showNum();
  delete r3;
  Rat::showNum();
  return 0;
}
```

Ratクラスのポインタ変数
r1,r2,r3を定義

静的メンバ



new演算子を用いてRatクラスのインスタンスを生成して、
そのアドレスをr1へ代入
(コンストラクタが実行される)

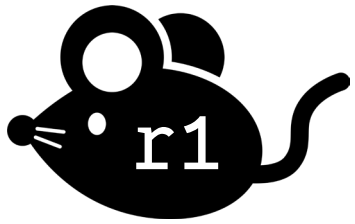
```
Rat* r1, * r2, * r3;
r1 = new Rat();
r1->squeak();
Rat::showNum();
r2 = new Rat();
r3 = new Rat();
r2->squeak();
r3->squeak();
```

```
Rat::showNum();
delete r1;
delete r2;
Rat::showNum();
```

コンストラクタ(r1)

```
Rat::Rat() : m_id(0) {
  s_count++;          //0 -> 1
  m_id = s_count; //1が代入
}
```

静的メンバ



- main.cpp（Sample405）

```
#include “rat.h”
```

静的メンバ関数**showNum()**

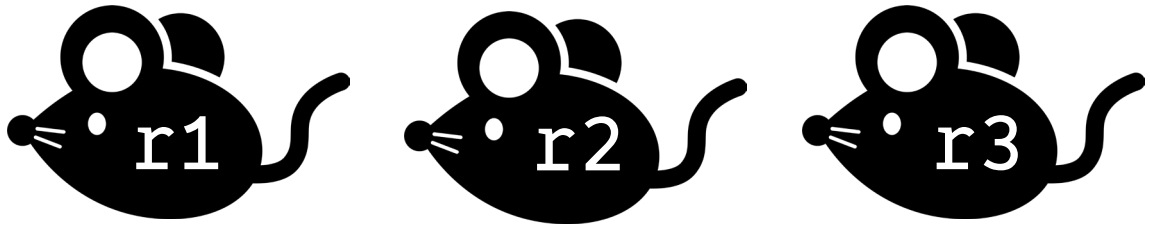
静的メンバ関数はインスタンス名を付けずに実行する！

```
Rat::showNum();
delete r1;
delete r2;
Rat::showNum();
delete r3;
```

```
    r1->squeak();
    Rat::showNum();
    r2 = new Rat();
    r3 = new Rat();
    r2->squeak();
    r3->squeak();
```

Rat::showNum()と記述することで、**静的メンバ関数**だということを明示する

静的メンバ



- main.cpp（Sample405）

```
#include “rat.h”

   Rat::showNum();
   delete r1;
   delete r2;
   r2 = new Rat();
   r3 = new Rat();
   r2->squeak();
   r3->squeak();
```

new演算子を用いてRatクラスのインスタンスを生成して、アドレスをr2,r3へ代入（コンストラクタが実行される）

コンストラクタ(r2)

```
Rat::Rat() : m_id(0) {
   s_count++;          //1 -> 2
   m_id = s_count; //2が代入
}
```

コンストラクタ(r3)

```
Rat::Rat() : m_id(0) {
   s_count++;          //2 -> 3
   m_id = s_count; //3が代入
}
```

静的メンバ

デストラクタ(r1)

```
Rat::~Rat() {
  cout << “ネズミ:” << m_id
    << “消去” << endl;
  s_count--;  //3 -> 2
}
```

デストラクタ(r2)

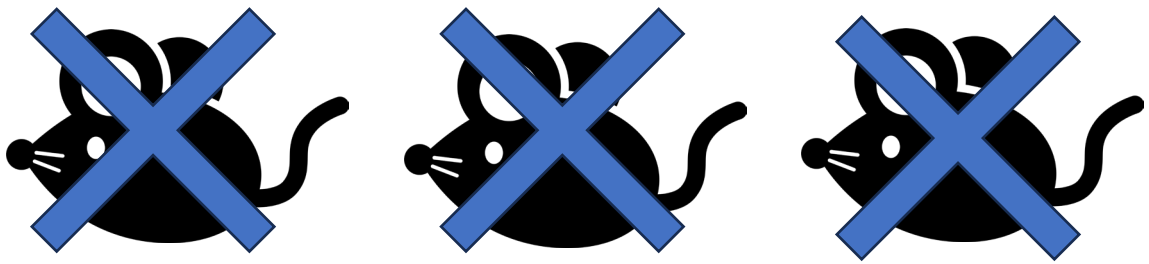
```
Rat::~Rat() {
  cout << “ネズミ:” << m_id
    << “消去” << endl;
  s_count--;  //2 -> 1
}
```

デストラクタ(r3)

```
Rat::~Rat() {
  cout << “ネズミ:” << m_id
    << “消去” << endl;
  s_count--;  //1 -> 0
}
```

delete演算子を用いて
r1,r2,r3を消去
(デストラクタが実行される)

```
  Rat::showNum();
  delete r1;
  delete r2;
  Rat::showNum();
  delete r3;
  Rat::showNum();
  return 0;
}
```



静的メンバ

- まとめ
  - **静的メンバ**は、インスタンスを生成せずにアクセス可能な変数や関数で、クラス自体が管理している
  - インスタンスメンバ（変数／関数）から静的メンバにアクセスは可能
  - 静的メンバからインスタンスメンバへのアクセスは不可（インスタンスが生成されていないことがあるため）