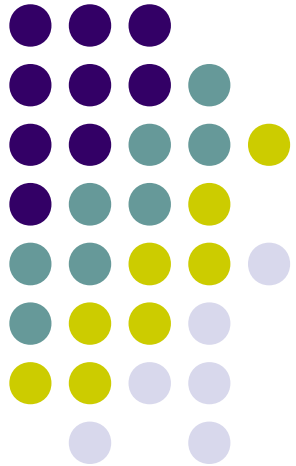
МОДЕЛЬ АСИНХРОННОГО СЕРЕДОВИЩА АГЕНТНИХ СИСТЕМ

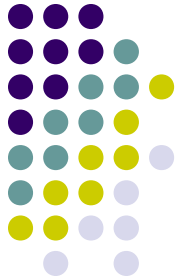
к.ф.-м.н. Нагорний В.А.

Друга Міжнародна науково-практична конференція

"Інформаційні технології та моделювання в економіці"



Поняття агентної моделі



- Агент - це інформаційна система, розташована в деякому *середовищі* і здатна до автономної дії в цьому середовищі щоб досягти свої визначені цілі.

- Автономна дія – здатність діяти без керуючого впливу: захист власного стану і поведінки від зовнішнього втручання

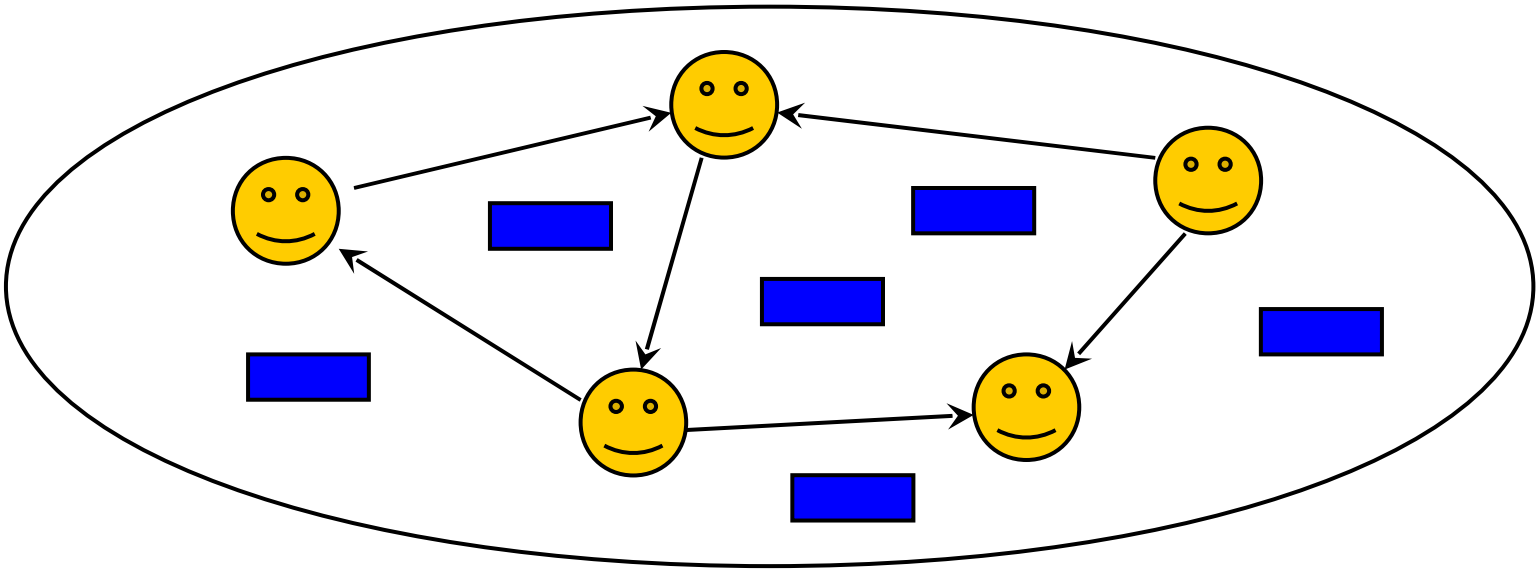
- Середовище – простір-посередник між іншими елементами моделі, в якому існують і взаємодіють як агенти так і ресурси

- Агентна модель = Середовище + Агенти + Ресурси

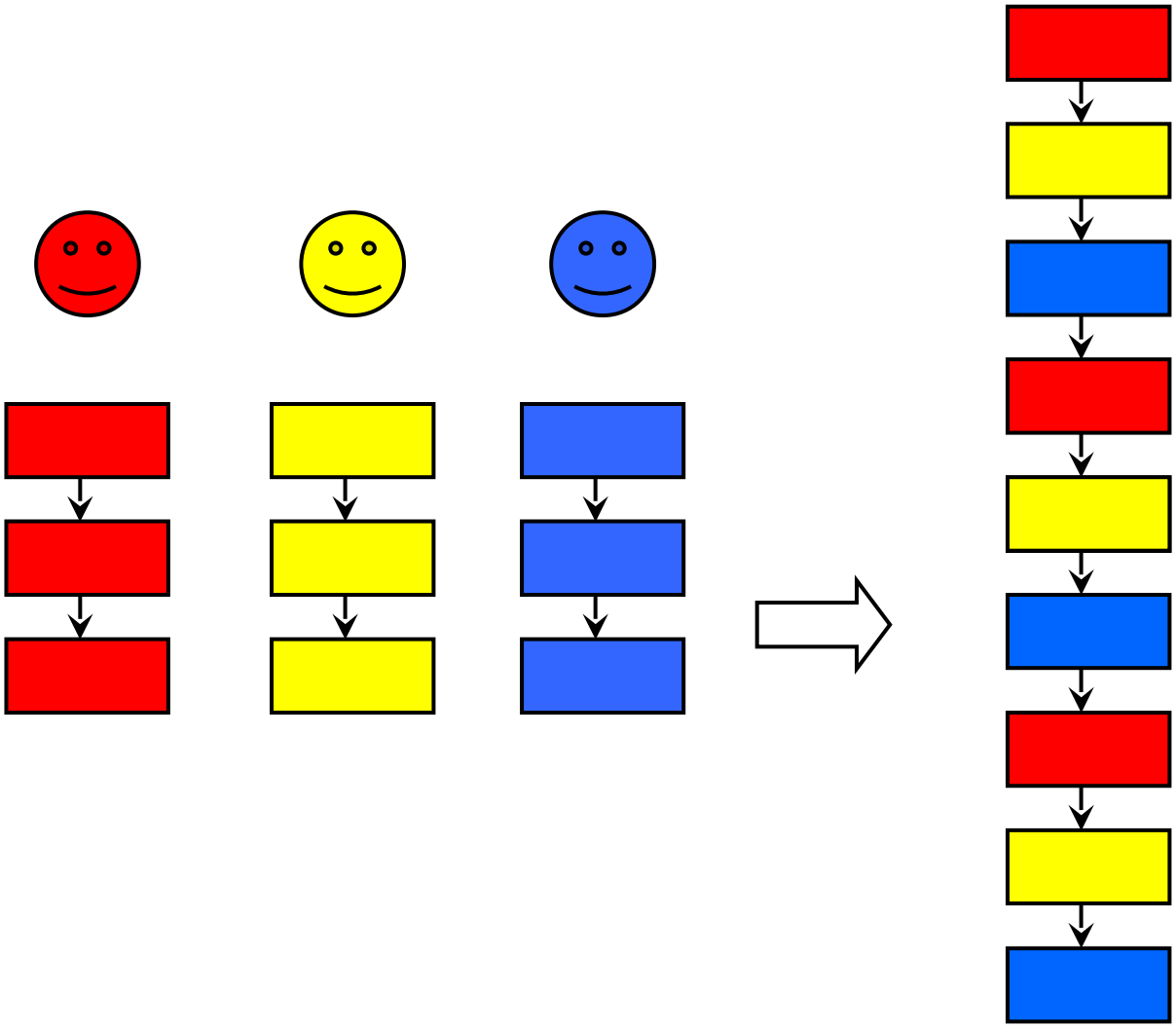
Вимоги до агентного середовища

- Забезпечити автономність поведінки агентів: незалежність та асинхронність дій агентів

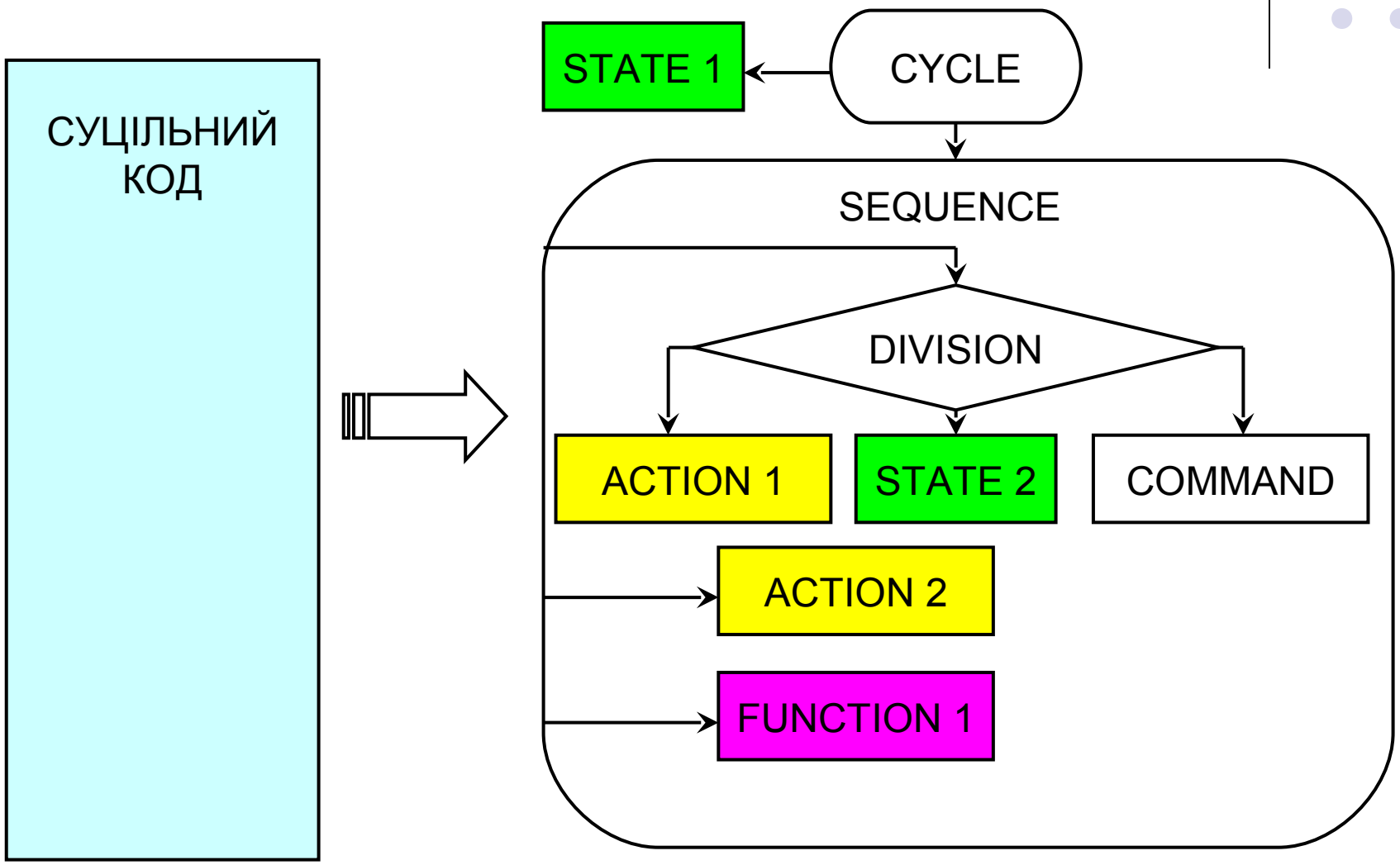
- Створити спільний простір для всіх елементів моделі, через який агенти могли б знаходити один одного та взаємодіяти, а також могли б отримувати доступ до інших ресурсів моделі



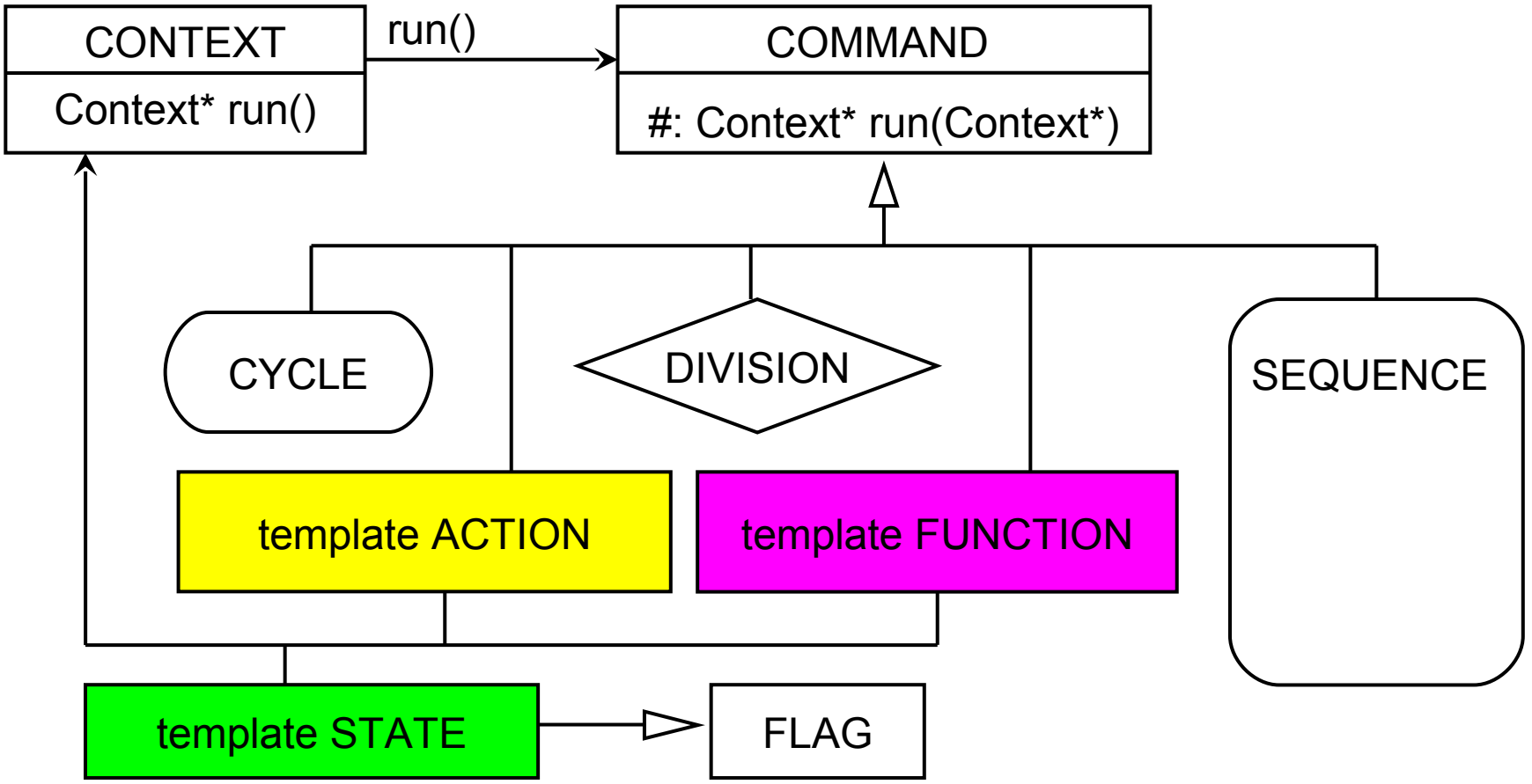
Асинхронізація поведінки



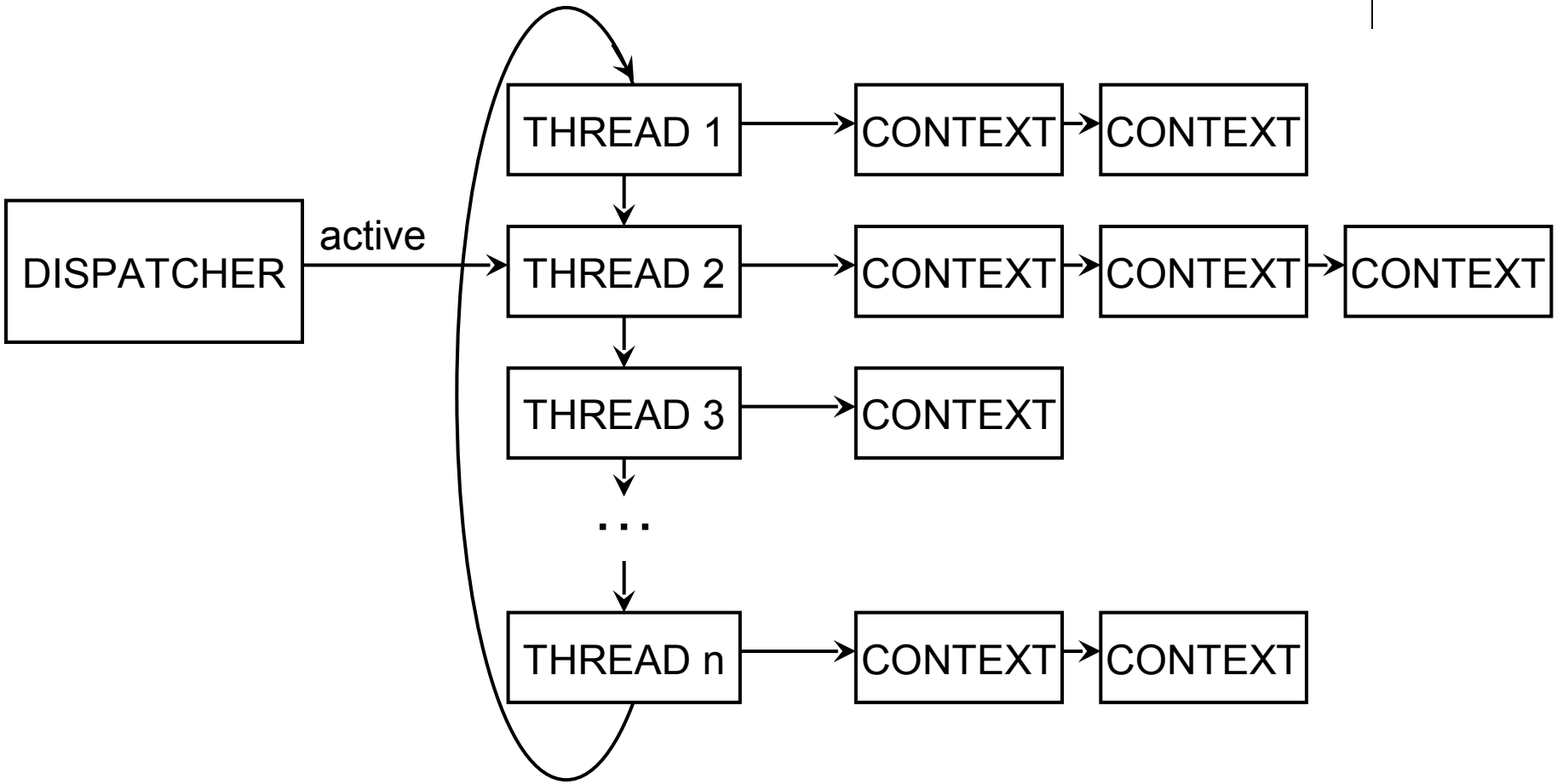
Алгоритм-об’єкт



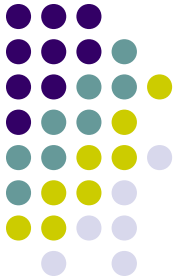
Діаграма класів складових алгоритму



Модель мультипаралельного виконання алгоритмів

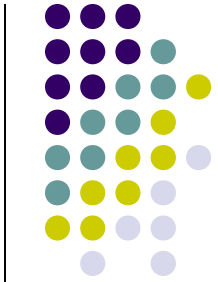


Приклад коду на C++



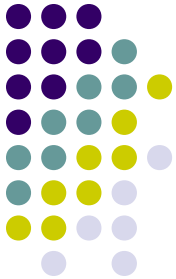
```
class Dispatcher { void operator+=(Context *C); bool run(); };

class Context { Context*  run(); };

class Command
{
            Command *prn, *nxt;
virtual     void next(Context* c);
virtual     Context* run(Context* c);
};

class Flag { virtual bool get(Context*); };

class Sequence : public Command
{
            Command *s,*f;
virtual     Context* run(Context* c);
            void operator +=(Command *c);
};
```

Розгалуження та цикл



```
class Division : public Command
{
            Command *on_true;
            Command *on_false;
            Flag    *flag;
virtual     Context* run(Context* c);
            void OnTrue(Command *c);
            void OnFalse(Command *c);
            void OnFlag(Flag *f);
};

class Cycle : public Command
{
            Command *on_true;
            Flag    *flag;
virtual     void next(Context*);
virtual     Context* run(Context *c);
            void OnTrue(Command *c);
            void OnFlag(Flag *f);
};
```

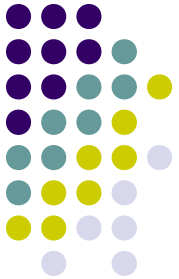
Операція, стан обчислень

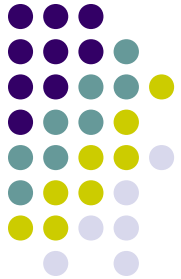


```
template<class C> class Action : public Command
{
        void (C::*a)(void);
        Action(void (C::*A)(void)):a(A){}
        virtual     Context* run(Context *c)
        {
                ((dynamic_cast<C*>(c))->*a)();
                next(c);
                return c;
        }
};

template<class C> class State : public Flag
{
        bool C::*f;
        State(bool C::*F):f(F){}
        virtual     bool get(Context* c)
        {
                return (dynamic_cast<C*>(c))->*f;
        }
};
```

Виклик функції

```
template<class C> class Function : public Command
{
        Context* C::*p;
        Function(Context* C::*P):p(P){}
        virtual      Context* run(Context *c)
        {
                next(c);
                return ((dynamic_cast<C*>(c))->*p)->run();
        }
        void OnContext(Context *c){p = c;}
};
```





приклад асинхронного середовища агентної моделі