ООП-2

1. Что представляет собой объект класса vector?

Ответ:
Динамический массив

2. Каким образом выделяется память для хранения элементов вектора?

Ответ:
Выделен один блок памяти. (Contiguous storage)

3. Какое из приведенных высказываний верно?

Ответ:
Значение элемента вектора можно получить с помощью смещения или указателя

4. Какое количество основных последовательных контейнеров определено в STL?

Ответ:
3

5. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
  unsigned int i;
  vector<int> first;
  vector<int> second (4, 100);
  vector<int> third (second.begin(), second.end());
  vector<int> fourth (third);
  int myints[] = {16, 2, 77, 29};
  vector<int> fifth (myints, myints + sizeof(myints) / sizeof(int) );
  for (vector<int> :: iterator it = fifth.begin(); it != fifth.end(); ++it)
       cout << ' ' << *it;
  return 0;
}
```

Ответ:
16 2 77 29

6. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <vector>
int main ()
{
   vector<int> myvector;
   int sum (0);
   myvector.push_back (100);
   myvector.push_back (200);
   myvector.push_back (300);
   while (!myvector.empty())
   {
      sum += myvector.back();
      myvector.pop_back();
   }
   cout << sum << '\n';
   return 0;
}
```

Ответ:
600

7. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
   vector<int> a (3, 0);
   vector<int> b (5, 0);
   b = a;
   a = vector<int>();
   cout << "Size of a= " << int(a.size());
   cout << " b= " << int(b.size()) << '\n';
   return 0;
}
```

Ответ:
Size of a= 0 b= 3

8. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
   vector<int> first;
   first.assign (7,100);
   vector<int>::iterator it;
   it=first.begin()+1;
   int myints[] = {1776,7,4};
   cout << int (first.size()) << '\n';
   return 0;
}
```

Ответ:
7

9. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
   vector<int> myvector (5);
   int* p = myvector.data();
   *p = 10;
   ++p;
   *p = 20;
   p[2] = 100;
   for (unsigned i = 0; i < myvector.size(); ++i)
       cout << ' ' << myvector[i];
   return 0;
}
```

Ответ:
10 20 0100 0

10. Выберете корректное определение вектора.

Ответ:
vector<int> vals(5);

11. Что можно опустить при определении вектора?

Ответ:
Число элементов

12. Какие из шаблонов последовательных контейнеров реализованы на базе массива?

Ответ:
vector и dequeue

13. Какой из методов добавит новый элемент в конце контейнера?

Ответ:
push_back

14. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <deque>
using namespace std;
int main ()
{
   deque<int> mydeque (5);
   deque<int>::reverse_iterator rit = mydeque.rbegin();
   int i = 0;
   for (rit = mydeque.rbegin(); rit!= mydeque.rend(); ++rit)
        *rit = ++i;
   for (deque<int>::iterator it = mydeque.begin(); it != mydeque.end(); ++it)
        cout << ' ' << *it;
   return 0;
}
```

Ответ:
5 4 3 2 1

15. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <deque>
using namespace std;
int main ()
{
   unsigned int i;
   deque<int> a (3,100);
   deque<int> b (5,200);
   a.swap(b);
   cout << "a contains:";
   for (deque<int>::iterator it = a.begin(); it != a.end(); ++it)
        cout << ' ' << *it;
   cout << " b contains:";
   for (deque<int>::iterator it = b.begin(); it != b.end(); ++it)
        cout << ' ' << *it;
   return 0;
}
```

Ответ:
a contains 200200200200 200 b contains 100100100

16. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <deque>
using namespace std;
int main ()
{
    unsigned int i;
    deque<int> mydeque;
    mydeque.push_back (100);
    mydeque.push_back (200);
    mydeque.push_back (300);
    for(deque<int> :: iterator it = mydeque.begin(); it != mydeque.end(); ++it)
    {
    }
    mydeque.clear();
    mydeque.push_back (110);
    mydeque.push_back (220);
    for(deque<int> :: iterator it = mydeque.begin(); it != mydeque.end(); ++it)
        cout << ' ' << *it;
    cout << '\n';
    return 0;
}
```

Ответ:
110 220

17. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
   vector<int> myvector;
   int * p;
   unsigned int i;
   p = myvector.get_allocator().allocate(5);
   for(i = 0; i < 5; i++)
        myvector.get_allocator().construct(&p[i], i);
   for(i = 0; i < 5; i++)
        cout << ' ' << p[i];
   for(i = 0; i < 5; i++)
        myvector.get_allocator().destroy(&p[i]);
   myvector.get_allocator().deallocate(p, 5);
   return 0;
}
```

Ответ:
0 1 2 3 4

18. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <cmath>
#include <list>
using namespace std;
bool same_integral_part (double first, double second)
{  return ( int(first) == int(second) );  }

struct is_near
{
   bool operator() (double first, double second)
   {  return (fabs(first - second) < 5.0);   }
};

int main ()
{
   double mydoubles[] =
   { 12.15, 2.72, 73.0, 12.77, 3.14, 12.77, 73.35, 72.25, 15.3, 72.25 };
   list<double> mylist (mydoubles, mydoubles + 10);
   mylist.sort();
   mylist.unique();
   mylist.unique (same_integral_part);
   mylist.unique (is_near());
   for (list<double> :: iterator it = mylist.begin(); it != mylist.end(); ++it)
       cout << ' ' << *it;
   cout << '\n';
   return 0;
}
```

Ответ:
2.72 12.15 72.25

19. Каким образом организован контейнер list?

Ответ:
Как двусвязный список

20. Какой из приведенных контейнеров поддерживает любые операции вставки и удаления элемента?

Ответ:
array

21. Какой интерфейс требуется в контейнере для управления памятью?

Ответ:
Allocator interface

22. Что определено в базовом классе Allocator interface?

Ответ:
Все перечисленное

23. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <vector>
using namespace std;
class Component
{ public:
    virtual void traverse() = 0;
};

class Leaf: public Component
{   int value;
  public:
    Leaf(int val): value(val){}
    void traverse()
    {  cout << value << ' '; }
};

class Composite: public Component
{   vector < Component * > children;
  public:
    void add(Component *ele)
    {   children.push_back(ele); }

    void traverse()
    {   for (int i = 0; i < children.size(); i++)
          children[i]->traverse();
    }
};

int main()
{  Composite containers[4];
   for (int i = 0; i < 4; i++)
       for (int j = 0; j < 3; j++)
           containers[i].add(new Leaf(i *3+j));
   for (int k = 1; k < 4; k++)
       containers[0].add(&(containers[k]));
   for (int p = 0; p < 4; p++)
   {   containers[p].traverse(); cout<<endl; }
}
```

Ответ:
ничего из приведенного

24. Итераторы какого из контейнеров поддерживают операцию произвольного доступа operator[] ?

Ответ:
vector

dequeue

25. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <vector>
#include <iterator>
#include <stddef.h>
using namespace std;

template<class myType> class SimpleContainer
{ public:
   SimpleContainer(size_t xDim, size_t yDim, myType const& defaultValue):
      objectData(xDim * yDim, defaultValue) , xSize(xDim) , ySize(yDim){}

   myType& operator()(size_t x, size_t y)
   {  return objectData[y * xSize + x];  }

   myType const& operator()(size_t x, size_t y) const
   {  return objectData[y * xSize + x];  }

   int getSize()
   {  return objectData.size();  }

   void inputEntireVector(vector<myType> inputVector)
   {  objectData.swap(inputVector);      }

   void printContainer(ostream& stream)
   {  copy(objectData.begin(), objectData.end(),
      ostream_iterator<myType>(stream, ""/*No Space*/));
   }
  private:
   vector<myType> objectData;
   size_t  xSize;
   size_t  ySize;
};

template<class myType> inline ostream& operator<<(
       ostream& stream, SimpleContainer<myType>& object)
{   object.printContainer(stream);
    return stream;
}

void sampleContainerInterfacing();
```

```
int main()
{  sampleContainerInterfacing();
   return 0;
}

void sampleContainerInterfacing()
{
   static int const ConsoleWidth  = 80;
   static int const ConsoleHeight = 25;
   size_t width  = ConsoleWidth;
   size_t height = ConsoleHeight;
   SimpleContainer<int> mySimpleContainer(width, height, 0);
   cout << mySimpleContainer.getSize() << endl;
   mySimpleContainer(0, 0) = 5;
}
```

Ответ:
Результат зависит от компилятора

26. При написании lookup table (справочной таблицы) какой контейнер целесообразно применить?

Ответ:
std::map

27. Какой из перечисленных типов необходимо определить в классе контейнере?

Ответ:
Тип итератор

28. Какие параметры обеспечивают доступ к содержимому буфера?

Ответ:
Начало и завершение (конец)

29. Выберете заголовочный файл в котором не определен класс строки.

Ответ:
<ios>

30. При работе с STL какое из утверждений относительно стиля программирования верно?

Ответ:
Все перечисленное.

31. Последовательный контейнер vector это:

Ответ:
Последовательный контейнер со скрытым динамическим массивом

32. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <vector>
using namespace std;
int main():q
{
  vector<int> v;
  v.assign( 10, 42 );
  for (int i = 0; i < v.size(); i++){
     cout << v[i] << " ";
  }
}
```

Ответ:
42 десять раз

33. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <list>
#include <queue>
using namespace std;
int main()
{
  queue<char> q;
  q.push('a');
  q.push('b');
  q.push('c');
  cout << q.front();
  q.pop();
  cout << q.front();
  q.pop();
  cout << q.front();
  q.pop();
}
```

Ответ:
abc

34. Что будет результатом выполнения следующей программы?

```
#include <list>
#include <string>
#include <iostream>
using namespace std ;
typedef list<string> LISTSTR;
int main()
{
  LISTSTR :: iterator i;
  LISTSTR test;
  test.insert(test.end(), "one");
  test.insert(test.end(), "two");
  LISTSTR test2(test);
  LISTSTR test3(3, "three");
  LISTSTR test4(++test3.begin(),
  test3.end());
  cout << "test:";
  for (i =  test.begin(); i != test.end(); ++i)
      cout << " " << *i << endl;
  cout << "test:";
  for (i =  test2.begin(); i != test2.end(); ++i)
      cout << " " << *i << endl;
  cout << "test:";
  for (i =  test3.begin(); i != test3.end(); ++i)
      cout << " " << *i << endl;
  cout << "test:";
  for (i =  test4.begin(); i != test4.end(); ++i)
      cout << " " << *i << endl;
}
```

Ответ:
Ничего из перечисленного

35. Выберете заголовочный файл, которого не существует?

Ответ:
<containers >

36. Что включает в себя понятие библиотека STL в C++?

Ответ:
Коллекция классов контейнеров и средств доступа к их содержимому и набор обобщенных алгоритмов

37. Выберете метод который переопределен для контейнера unordered_set в C++.

Ответ:
count

38. Что воплощают ассоциативные контейнеры?

Ответ:
Ассоциативные массивы

39. Сколько типов ассоциативных контейнеров в STL?

Ответ:
5

40. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <string>
#include <bitset>
using namespace std;
int main ()
{
   string mystring;
   bitset<4> mybits;
   mybits.set();
   mystring = mybits.to_string<char, char_traits<char>,
   allocator<char> >();
   cout << mystring << endl;
   return 0;
}
```

Ответ:
1111

41. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main ()
{
  vector<int> first (5, 10);
  vector<int> second (5, 33);
  vector<int>::iterator it;
  swap_ranges(first.begin() + 1, first.end() - 1, second.begin());
  cout << " first=";
  for (it = first.begin(); it != first.end(); ++it)
       cout << " " << *it;
  cout << " second=";
  for (it = second.begin(); it != second.end(); ++it)
       cout << " " << *it;
  return 0;
}
```

Ответ:
first= 10 33 33 33 10 second= 10 10 10 33 33

42. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <map>
using namespace std;
int main ()
{
   map<char, int> mymap;
   map<char, int> :: iterator it;
   mymap['b'] = 100;
   mymap['a'] = 200;
   mymap['c'] = 300;
   for (map<char, int>::iterator it = mymap.begin(); it != mymap.end(); ++it)
       cout << it->first << " => " << it->second << " | ";
   cout << '\n';
   return 0;
}
```

Ответ:
a => 200 | b => 100 | c => 300 |

43. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <set>
using namespace std;
int main ()
{
  set<int> myset;
  myset.insert(20);
  myset.insert(30);
  myset.insert(10);
  while (!myset.empty())
  {
      cout << ' ' << *myset.begin();
      myset.erase(myset.begin());
  }
  cout << '\n';
  return 0;
}
```

Ответ:
10 20 30

44. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <set>
using namespace std;
int main ()
{
   multiset<int> mymultiset;
   for (int i = 0; i < 5; i++) mymultiset.insert(i);
   multiset<int> :: key_compare mycomp = mymultiset.key_comp();
   int highest = *mymultiset.rbegin();
   multiset<int> :: iterator it = mymultiset.begin();
   do{
        cout << ' ' << *it;
   }while(mycomp(*it++, highest));
   return 0;
}
```

Ответ:
0 1 2 3 4

45. Сколько одинаковых экземпляров объекта допускается при записи в объект классов map или set?

Ответ:
1

46. Какие итераторы поддерживают классы map и set?

Ответ:
Двунаправленные итераторы

47. Как можно охарактеризовать тип библиотеки STL?

Ответ:
Обобщенный

48. Элементами какого типа может быть инстанцирован (специализирован) контейнер STL?

Ответ:
Любым типом

49. Как можно описать тип контейнера list?

Ответ:
Node-based

50. Какой тип доступа обеспечивают контейнеры vector и dequeue?

Ответ:
Произвольный доступ

51. Наиболее эффективная операция добавления элемента в вектор при вставке элемента в …

Ответ:
Конец

52. Какие из перечисленных классов не являются полноценными контейнерными классами?

Ответ:
Адаптеры

53. Чем ограничено время жизни элемента контейнера?

Ответ:
Временем жизни контейнера

54. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <map>
using namespace std;
int main ()
{
  multimap<char, int> mymultimap;
  mymultimap.insert(make_pair('x', 100));
  mymultimap.insert(make_pair('y', 200));
  mymultimap.insert(make_pair('y', 350));
  mymultimap.insert(make_pair('z', 500));
  cout << mymultimap.size() << '\n';
  return 0;
}
```

Ответ:
4

55. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <queue>
using namespace std;
int main ()
{
   priority_queue<int> mypq;
   mypq.push(10);
   mypq.push(20);
   mypq.push(15);
   cout  << mypq.top() << endl;
   return 0;
}
```

Ответ:
20

56. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <map>
using namespace std;
int main ()
{
  multimap<char, int> mymultimap;
  mymultimap.insert(make_pair('y', 202));
  mymultimap.insert(make_pair('y', 252));
  pair<char, int> highest = *mymultimap.rbegin();
  multimap<char, int> :: iterator it = mymultimap.begin();
  do{
     cout << (*it).first << " => " << (*it).second << " ";
  }while( mymultimap.value_comp()(*it++, highest) );
  return 0;
}
```

Ответ:
y => 202

57. Что определено для каждого контейнерного класса?

Ответ:
Все перечисленное

58. Что мы вернем, если используем обычный массив в качестве внутреннего контейнера?

Ответ:
Указатель

59. Что должен содержать проектируемый новый контейнера?

Ответ:
Итераторы

60. Какое количество итераторов требуется при определении контейнера?

Ответ:
3

61. Каково назначение класса allocator в пользовательском контейнере?

Ответ:
Ничего из перечисленного

62. Как называется контейнер, который содержит группу разных объектов?

Ответ:
Гетерогенный контейнер

63. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int main()
{
  string s = "spaces in text";
  s.erase(remove(s.begin(), s.end(), ' ' ), s.end() ) ;
  cout << s << endl;
}
```

Ответ:
spacesintext

64. Что будет результатом выполнения следующей программы?

```
#include <vector>
#include <algorithm>
#include <iostream>
#include <iterator>
using namespace std;
int square(int i) { return i * i; }
int main()
{
  vector<int> V, V2;
  V.push_back(0);
  V.push_back(1);
  V.push_back(2);
  transform(V.begin(), V.end(), back_inserter(V2), square);
  copy(V2.begin(), V2.end(), ostream_iterator<int>(cout, " "));
  cout << endl;
}
```

Ответ:
0 1 4

65. Какой заголовочный файл требуется для работы с алгоритмами STL?

Ответ:
algorithm

66. Выберете правильный метод STL.

Ответ:
mismatch

67. Какое ключевое слово используется для перегрузки оператора?

Ответ:
operator

68. Для чего предназначен метод make_heap в операциях с кучей?

Ответ:
Form heap – построить кучу

69. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main ()
{
  int first[] = {5, 10, 15, 20, 25};
  int second[] = {50, 40, 30, 20, 10};
  vector<int> v(10);
  vector<int> :: iterator it;
  sort (first, first + 5);
  sort (second, second + 5);
  it = set_union (first, first + 5, second, second + 5, v.begin());
  cout << int(it - v.begin());
  return 0;
}
```

Ответ:
8

70. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main ()
{
  vector<int> myvector (4);
  fill (myvector.begin(), myvector.begin() + 2, 3);
  fill (myvector.begin() + 1, myvector.end() - 1, 4);
  for (vector<int> :: iterator it = myvector.begin(); it != myvector.end(); ++
      cout << ' ' << *it;
  return 0;
}
```

Ответ:
3 4 4 0

71. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main ()
{
  vector<int> myvector;
  for (int i = 1; i < 6; ++i)
      myvector.push_back(i);
  reverse(myvector.begin(), myvector.end());
  for (vector<int> :: iterator it = myvector.begin(); it != myvector.end(); ++it)
      cout << ' ' << *it;
  return 0;
}
```

Ответ:
5 4 3 2 1

72. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main ()
{
  vector<int> myvector;
  for (int i = 1; i < 6; ++i)
      myvector.push_back(i);
  reverse(myvector.begin(), myvector.end());
  for (vector<int> :: iterator it = myvector.begin(); it != myvector.end(); ++it)
      cout << ' ' << *it;
  return 0;
}
```

Ответ:
5 4 3 2 1

73. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main ()
{
  int myints[] = {10, 20, 30, 30, 20, 10, 10, 20};
  int mycount = count (myints, myints + 8, 10);
  cout << " " << mycount;
  vector<int> myvector (myints, myints+8);
  mycount = count (myvector.begin(), myvector.end(), 20);
  cout << " " << mycount << endl
  return 0;
}
```

Ответ:
3 3

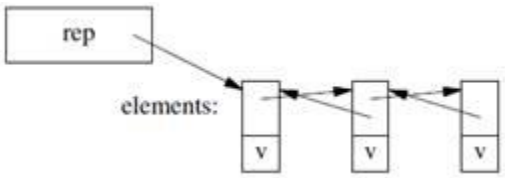
74. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <algorithm>
using namespace std;
int main ()
{
  int myints[] = {10, 20, 30, 30, 20, 10, 10, 20};
  int* pbegin = myints;
  int* pend = myints + sizeof(myints)/sizeof(int);
  pend = remove (pbegin, pend, 20);
  for (int* p = pbegin; p != pend; ++p)
      cout << ' ' << *p;
  return 0;
}
```

Ответ:
10 30 30 10 10

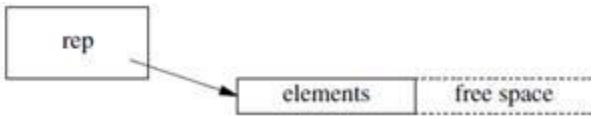
75. Какая из приведенных схем распределения памяти соответствует контейнеру list?

Ответ:



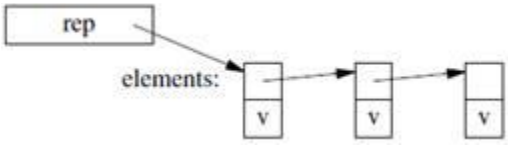
76. Какая из приведенных схем распределения памяти соответствует контейнеру vector?

Ответ:



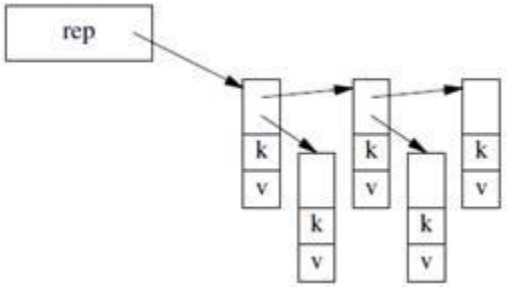
77. Какая из приведенных схем распределения памяти соответствует контейнеру forward_list?

Ответ:



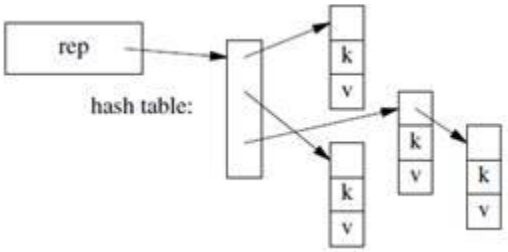
78. Какая из приведенных схем распределения памяти соответствует контейнеру map?

Ответ:



79. Какая из приведенных схем распределения памяти соответствует контейнеру unordered_map?

Ответ:



80. Контейнеры это:

Ответ:
Шаблонные классы, которые настраиваются на работу с определенным типом

81. Контейнеры могут содержать элементы:

Ответ:
Только одного типа – однородные коллекции

82. Для создания неоднородной коллекции объектов

Ответ:
Требуется создать иерархию классов и хранить элементы базового класса

83. Упорядоченный, и неупорядоченный тип контейнера относится к

Ответ:
Ассоциативным контейнерам

84. Контейнеры которые, поддерживают последовательное (непрерывное) распределение памяти обеспечивают:

Ответ:
Быстрый доступ, но операции вставки и удаления элемента выполняются медленно

85. Контейнеры которые, поддерживают связанное распределение памяти обеспечивают:

Ответ:
Быстрые операции вставки и удаления элемента и медленный доступ к элементу

86. Используйте последовательный контейнер _____, когда нужна частая вставка/удаление и редкий поиск.

Ответ:
list

87. Какой из приведенных контейнеров обеспечит операции поиска и доступа O(n), вставку и удаление элемента O(1)?

Ответ:
list

88. Какой контейнер обеспечивает быстрый доступ к элементу, вставку/удаление на концах со сложностью алгоритма O(1)?

Ответ:
dequeue

89. Используйте последовательный контейнер _____, когда надо вставлять/удалять элементы на концах контейнера, и нужен быстрый доступ к элементу.

Ответ:
dequeue

90. Какой контейнер рекомендуют использовать вместо массивов обеспечивает быстрый доступ к элементу, вставку/удаление в конец контейнера со сложностью алгоритма O(1)?

Ответ:
vector

91. Используйте последовательный контейнер _____, когда надо вставлять/удалять элементы в конец контейнера, и нужен быстрый доступ к элементу.

Ответ:
vector

92. К какому типу относится контейнер, который реализует дисциплину обслуживания FIFO –First In First Out, вставку элемента на одном конце контейнера, а удаление на другом O(1)?

Ответ:
Адаптер

93. К какому типу относится контейнер priority_queue (очередь с приоритетами)?

Ответ:
Адаптер

94. Какой вариант реализации из перечисленных относится к контейнеру stack?

Ответ:
Дисциплина обслуживания last-in, first-out (LIFO)

95. Какой вариант реализации из перечисленных относится к контейнеру queue?

Ответ:
Реализует дисциплину обслуживания FIFO –First In First Out

96. Как называется контейнер, который реализует дисциплину обслуживания FIFO – First In First Out, вставку элемента на одном конце контейнера, а удаление на другом O(1)?

Ответ:
queue

97. Какой тип контейнера следует выбрать, если требуется одинаковое время на операции вставки, удаления, доступа к элементу?

Ответ:
Ассоциативный

98. Какой из приведенных контейнеров обеспечит время на операции вставки, удаления, доступа к элементу O(log(n))?

Ответ:
set

map

multiset

multimap

99. Какой из перечисленных итераторов не поддерживает операции сравнения (==), и нет оператора разыменования (->)?

Ответ:
Output (Write)

100. Какой из перечисленных итераторов является эквивалентом обычного указателя и поддерживает арифметику указателей, оператор выборки элемента из массива, а также все виды операций сравнения?

Ответ:
Random Access

101. Какие из перечисленных операций поддерживает итератор Forward?

Ответ:
operator++

operator*

operator->

operator==

102. Какие из перечисленных операций поддерживает итератор Input (Read)?

Ответ:
operator++

operator*

operator->

operator==

operator=

103. Какие из перечисленных операций поддерживает итератор Bidirectional?

Ответ:
operator--

operator->

operator++

operator=

104. Какие из перечисленных операций поддерживает итератор Output (Write)?

Ответ:
operator++

operator=

operator*

105. Классы vector, deque и list реализованы на базе массивов?

Ответ:
Ложь

106. Все ассоциативные контейнеры за исключением multiset поддерживают (bidirectional iterators) двунаправленные итераторы?

Ответ:
Ложь

107. STL реализованы как функции члены STLконтейнеров?

Ответ:
Ложь

108. Указатели в массивах можно использовать вместо итераторов для большинства STL алгоритмов, включая те, которые требуют итераторов с произвольным доступом?

Ответ:
Ложь

109. STL в основном исключает обобщенное программирование в пользу наследования и полиморфизма для достижения лучшего быстродействия?

Ответ:
Ложь

110. STL контейнер, который содержит элементы в определенном порядке , и где множественные элементы могут иметь одинаковые значения называется _____.

Ответ:
multiset

111. STL контейнер, который содержит уникальные элементы следующие в определенном порядке, называется _____.

Ответ:
set

112. STL контейнер, который содержит составные элементы ключ и значение, следующие в определенном порядке , называется _____.

Ответ:
map

113. STL контейнер, который содержит составные элементы ключ и значение, следующие в определенном порядке, и где множественные элементы могут иметь одинаковые ключи, называется _____.

Ответ:
multimap

114. Итератор позволяет выполнить операцию вставки и удаления элемента либо последовательности элементов в любом месте контейнера?

Ответ:
Истина

115. Итератор позволяет использовать алгоритмы STL?

Ответ:
Истина

116. Применение итераторов для доступа к элементам контейнера зачастую более эффективно относительно операции индексации элемента контейнера?

Ответ:
Истина

117. При сравнении итераторов чтения (input iterator) , какие допустимо использовать операции?

Ответ:
==, !=

118. Какие из приведенных операций, поддерживают все типы контейнеров библиотеки STL?

Ответ:
копирующий конструктор

метод проверки контейнера на пустоту empty()

конструктор по умолчанию

метод size(), который вернет число элементов в контейнере

119. Что бы получить итератор на первый элемент контейнера Cnt используют функцию ___?

Ответ:
Cnt.begin();

120. Что бы получить итератор на ппоследний элемент контейнера Cnt используют функцию ___?

Ответ:
Cnt.end();

121. Что бы объявить итератор it на контейнер vector целого типа нужно применить ___?

Ответ:
vector<int>::iterator it;

122. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <vector>
#include <iterator>
#include <algorithm>
using namespace std;

int main()
{
    vector<int> vec;
    vector<int>::iterator pos;
    vec.push_back(7); vec.push_back(4);  vec.push_back(8);
    vec.push_back(0); vec.push_back(12); vec.push_back(9);
    for(pos = vec.begin(); pos != vec.end(); pos++)
        cout<<*pos<< " ";
    pos = min_element(vec.begin(), vec.end());
    cout << " elem" << distance(vec.begin(), pos) << " " << *pos << endl;
    return 0;
}
```

Ответ:
7 4 8 0 12 9 elem3 0

123. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <vector>
#include <iterator>
#include <algorithm>
using namespace std;

int main()
{
    vector<int> vec;
    vector<int>::iterator pos;
    vec.push_back(7); vec.push_back(4);  vec.push_back(8);
    vec.push_back(0); vec.push_back(12); vec.push_back(9);
    for(pos = vec.begin(); pos != vec.end(); pos++)
        cout<<*pos<< " ";
    pos = max_element(vec.begin(), vec.end());
    cout << " elem" << distance(vec.begin(), pos) << " " << *pos << endl;
    return 0;
}
```

Ответ:
7 4 8 0 12 9 elem4 12

124. Что будет результатом выполнения следующей программы?

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

int main()
{
    vector<int> vec;
    vector<int>::iterator pos;
    vec.push_back(7); vec.push_back(4);  vec.push_back(8);
    vec.push_back(0); vec.push_back(12); vec.push_back(9);
    sort(vec.begin(), vec.end());
    for(pos = vec.begin(); pos != vec.end(); pos++)
        cout<<*pos<< " ";
    return 0;
}
```

Ответ:
0 4 7 8 9 12

125. Что будет результатом выполнения следующей программы?

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

int main()
{
    vector<int> vec;
    vector<int>::iterator pos;
    vec.push_back(7); vec.push_back(4);  vec.push_back(8);
    vec.push_back(0); vec.push_back(12); vec.push_back(9);
    pos = find(vec.begin(), vec.end(), 8);
    reverse(pos, vec.end());
    for(pos=vec.begin(); pos!=vec.end(); ++pos)
        cout<<*pos<<' ';
    cout << endl;
    return 0;
}
```

Ответ:
7 4 9 12 0 8

126. Что будет результатом выполнения следующей программы?

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

int main()
{
    vector<int> vec;
    vector<int>::iterator pos;
    vec.push_back(7); vec.push_back(4);  vec.push_back(8);
    vec.push_back(0); vec.push_back(12); vec.push_back(9);
    sort(vec.begin(), vec.end());
    reverse(vec.begin(), vec.end());
    for(pos = vec.begin(); pos != vec.end(); pos++)
        cout<<*pos<< " ";
    return 0;
}
```

Ответ:
12 9 8 7 4 0

127. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <vector>
#include <list>
#include <algorithm>
using namespace std;

int main()
{
    list<int> lst1;
    vector<int> vec1;
    list<int>::iterator pos;
    for(int i=1; i<=9; ++i)
        lst1.push_back(i);
    //vec1.resize(lst1.size());
    copy (lst1.begin(), lst1.end(), vec1.begin());
    for(pos = lst1.begin(); pos != lst1.end(); pos++)
        cout<<*pos<< " ";
    return 0;
}
```

Ответ:
Run time error

128. Применение функции члена контейнера предпочтительнее применения алгоритма, выполняющего аналогичную операцию? Например метод remove() для контейнера list и алгоритм remove().

Ответ:
Истина

129. Функция предикат это специальная вспомогательная функция, для алгоритмов, которая возвращает тип _____.

Ответ:
bool

130. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <list>
#include <algorithm>
#include <cstdlib>
using namespace std;

// a predicate, вернет истина для простого числа
bool isPrimeNum(int number)
{   number = abs(number);           // игнорируем знак
    if(number == 0 || number == 1)  // 0 и 1 простые числа
    {   return true;   }

    int divisor;    // поиск делителя на который делимое делится без остатка
    for(divisor = (number/2); (number%divisor) != 0; --divisor){;}

    return divisor == 1; // если делитель равен 1 - число простое
}

int main()
{
    list<int> lst1;
    for(int i=10; i<=18; ++i)
    lst1.push_back(i);
    list<int>::iterator pos = lst1.begin();
    pos++; pos++;
    pos = find_if(pos, lst1.end(), isPrimeNum);
    if(pos != lst1.end()){
        cout<<*pos<<endl;
    }else{
        cout<<"X"<<endl;
    }
    return 0;
}
```

Ответ:
13

131. Сложность алгоритмов описывается O- нотацией. Например: O(n) –линейная, O(log(n)) – логарифмическая. Значит ли это что для разового вычисления время работы алгоритма соответствует O-нотации, если O(n2)?

Ответ:
Нет

132. Следует ли ожидать от алгоритма заявленного быстродействия, если при выполнении операции происходит перераспределение памяти?

Ответ:
Нет

133. Что будет результатом выполнения следующей программы?

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

int main()
{
    list<int> lst;
    list<int>::iterator Iter;
    list<int>::iterator result1;
    lst.push_back(14);  lst.push_back(17);  lst.push_back(31);
    lst.push_back(31);  lst.push_back(10);  lst.push_back(20);

    result1 = adjacent_find(lst.begin(), lst.end());
    if(result1 == lst.end())
        cout<<" X"<<endl;
    else
        cout<<" "<<*(result1)<<endl;
    return 0;
}
```

Ответ:
31

134. Что будет результатом выполнения следующей программы?

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

int main()
{
    list<int> lst;
    list<int>::iterator Iter;
    list<int>::iterator result1;
    lst.push_back(14);  lst.push_back(17);  lst.push_back(31);
    lst.push_back(31);  lst.push_back(10);  lst.push_back(20);

    result1 = adjacent_find(lst.begin(), lst.end());
    if(result1 == lst.end())
        cout<<" X"<<endl;
    else
        cout<<" "<<*(result1)<<endl;
    return 0;
}
```

Ответ:
10 & 20

135. Что будет результатом выполнения следующей программы?

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

int main()
{
    vector <int> vec1, vec2;
    vector <int>::iterator Iter1, Iter2;
    for(int i = 0; i <= 5; i++)
        vec1.push_back(i);
    for(int j = 10; j <= 20; j++)
        vec2.push_back(j);

    copy(vec1.begin(), vec1.begin() + 4, vec2.begin() + 5);

    for(Iter2 = vec2.begin(); Iter2 != vec2.end(); Iter2++)
        cout<<*Iter2<<" ";
    cout<<endl;

    return 0;
}
```

Ответ:
10 11 12 13 14 0 1 2 3 19 20

136. Что будет результатом выполнения следующей программы?

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

int main()
{
    vector <int> vec1, vec2;
    vector <int>::iterator Iter1, Iter2;
    for(int i = 10; i <= 15; i++)
        vec1.push_back(i);
    for(int j = 0; j <= 10; j++)
        vec2.push_back(j);

    copy_backward(vec1.begin(), vec1.begin() + 4, vec2.begin() + 8);

    for(Iter2 = vec2.begin(); Iter2 != vec2.end(); Iter2++)
        cout<<*Iter2<<" ";
    cout<<endl;
    return 0;
}
```

Ответ:
0 1 2 3 10 11 12 13 8 9 10

137. Какой метод использует алгоритм STL count
template <class InputIterator, class Type> typename iterator_traits<InputIterator>::difference_type count(InputIterator _First, InputIterator _Last, const Type& _Val); ?

Ответ:
operator==

138. Что будет результатом выполнения следующей программы?

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

int main()
{
    vector <int> vec;
    vector <int>::iterator Iter;
    vec.push_back(12);  vec.push_back(22);  vec.push_back(12);
    vec.push_back(31);  vec.push_back(12);  vec.push_back(33);

    int result = count(vec.begin(), vec.end(), 12);

    cout<<"result is: "<<result<<endl;
    return 0;
}
```

Ответ:
result is: 3

139. Что будет результатом выполнения следующей программы?

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

bool isgreat(int value)
{    return value >8;    }

int main()
{
    vector <int> vec;
    vector <int>::iterator Iter;
    vec.push_back(13);  vec.push_back(21);  vec.push_back(9);
    vec.push_back(31);  vec.push_back(8);   vec.push_back(10);

    for(Iter = vec.begin(); Iter != vec.end(); Iter++)
        cout<<*Iter<<" ";

    int result1 = count_if(vec.begin(), vec.end(), isgreat);
    cout<<"result is: "<<result1<<endl;
    return 0;
}
```

Ответ:
13 21 9 31 8 10 result is: 5

140. Что будет результатом выполнения следующей программы?

```
#include <iostream>
#include <algorithm>
#include <functional>
#include <string>
#include <vector>
using namespace std;

// return true if string str starts with letter 'C'
int MatchFirstChar(const string& str)
{
    string s("C");
    return s == str.substr(0, 1);
}

int main()
{
    const int VECTOR_SIZE = 110;
    typedef vector<string > StringVector;
    typedef StringVector::iterator StringVectorIt;

    StringVector NamesVect(VECTOR_SIZE);
    StringVectorIt start, end, it;
    ptrdiff_t result = 0;

    NamesVect[0] = "Learn";          NamesVect[1] = "C";
    NamesVect[2] = "and";            NamesVect[3] = "C++";

    start = NamesVect.begin();
    end = NamesVect.end();

    result = count_if(start, end, MatchFirstChar);

    cout<<"result is: "<<result<<endl;
}
```

Ответ:
result is: 2

141. Что будет результатом выполнения следующей программы?

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

// return whether second element is twice of the first
bool twice(int elem1, int elem2)
{    return elem1 * 2 == elem2;    }

int main()
{
    vector <int> vec1, vec2, vec3;
    vector <int>::iterator Iter1, Iter2, Iter3;
    for(int i = 0; i <= 5; i++)
        vec1.push_back(i);
    for(int j = 0; j <= 5; j++)
        vec2.push_back(2*j);
    for(int k = 0; k <= 5; k++)
        vec3.push_back(k);

    bool b = equal(vec1.begin(), vec1.end(), vec3.begin());
    if(b)
        cout<<" vec1 EQ vec3 |";
    else
        cout<<" vec1 NOTEQ vec3 |";


    bool c = equal(vec1.begin(), vec1.begin()+5, vec2.begin(), twice);
    if(c)
        cout<<" vec1 EQ vec2 twice|";
    else
        cout<<" vec1 NOTEQ vec2 twice|";

    return 0;
}
```

Ответ:
vcc1 EQ vec3 | vec1 EQ vec2 twice |

142. Для работы с файлами C++ обеспечивает несколько _____ таких как ifstream, ofstream и fstream.

Ответ:
Классов

143. Ifstream открывает файл для _____, ofstream для _____ и fstream для _____.

Ответ:
Чтения, записи, чтения и записи

144. По умолчанию все файлы открываются в режиме _____.

Ответ:
Ввод (чтение)

Вывод (запись)

145. Функция get(chr) считывает _____ из _____, а put(chr) читает символ из _____ в _____.

Ответ:
Потока ввода, переменная chr, переменная chr поток вывода

146. Методы чтения (read), записи (write) предназначены для операций _____ из/в поток.

Ответ:
Ввода/вывода блока

147. Функция istream& getline (char* s, streamsize n);
определена в _____ и служит для _____.

Ответ:
Iostream, ввод n символов из потока и сохраняет их в виде строки

148. Какой из перечисленных классов обеспечивает ввод из файла?

Ответ:
ifstream

149. Какой из потоков класса iostream относится к стандартному не буферизированному потоку ошибок?

Ответ:
cerr

150. Какой из потоков класса iostream относится к стандартному буферизированному потоку ошибок?

Ответ:
clog

151. Какой из потоков класса iostream относится к стандартному потоку ввода?

Ответ:
cin

152. Какой из потоков класса iostream относится к стандартному потоку вывода?

Ответ:
cout

153. Какой заголовочный файл необходимо подключить для работы с потоками стандартного ввода вывода?

Ответ:
<iostream>

154. Какой заголовочный файл необходимо подключить для осуществления операций форматирования потока?

Ответ:
<iomanip>

155. Какой заголовочный файл необходимо подключить для работы с файлами?

Ответ:
<fstream>

156. Какой заголовочный файл необходимо подключить для выполнения операций форматирования в памяти, для работы с символьными массивами или строками?

Ответ:
<strstream>

157. Какой из перечисленных классов является родительским классом (предком) для класса iostream?

Ответ:
ostream

istream

158. Какой из перечисленных классов является родительским классом (предком) для класса ifstream?

Ответ:
istream

159. Какой из перечисленных классов является родительским классом (предком) для класса ofstream?

Ответ:
ostream

160. Какой из перечисленных классов является родительским классом (предком) для класса ostream?

Ответ:
iostream

161. Какой из перечисленных классов является родительским классом (предком) для класса istream?

Ответ:
iostream

162. Какой из перечисленных классов является родительским классом (предком) для класса fstream?

Ответ:
iostream

163. Какой из приведенных операторов называется оператором вставки в поток и предназначен для потока стандартного вывода?

Ответ:
<<

164. Какой из приведенных операторов называется оператором извлечения из потока и предназначен для потока стандартного ввода?

Ответ:
>>

165. Какой из вариантов функции get следует вызвать, если требуется прочитать строку символов из потока?

Ответ:
istream& get (char* s, streamsize n);

istream& get (char* s, streamsize n, char delim);

166. Какой из вариантов функции get следует вызвать, если требуется прочитать текущий символ из потока?

Ответ:
int get();

istream& get (char& c);

167. Какой из вариантов функции get будет вызван для приведенного кода?

```
char str[256];

std::cout << "Enter the name of an existing text file: ";
std::cin.get (str,256);
```

Ответ:
istream& get (char* s, streamsize n);

168. Какой из вариантов функции get будет вызван для приведенного кода?

```
std::ifstream is(str);

char c;
while (is.get(c))
  std::cout << c;
```

Ответ:
istream& get (char& c);

169. При вводе строки из потока допустимо пользоваться как функцией getline, так и функцией get?

Ответ:
Истина

170. При запуске программы, пользователь ввел имя и фамилию Вася Силин. Что выведет программа? Код представлен ниже.

```
#include <iostream>
int main () {
  char first, last;
  std::cout << "Введите Имя и Фамилию: ";
  first = std::cin.get();
  std::cin.ignore(256,' ');
  last = std::cin.get();
  std::cout << "Привет, " << first << last << "!\n";
  return 0;
}
```

Ответ:
Привет, ВС!

171. Что выведет на экран приведенный фрагмент кода /*Вариант А*/ и /*Вариант В*/, если пользователь ввел число 3456?

```
/*Вариант A*/
    char c = std::cin.get();
    int n;
    std::cin.putback (c);
    std::cin >> n;
    std::cout << " A: " << n;
....
/*Вариант B*/
    char c = std::cin.get();
    int n;
    std::cin >> n;
    std::cout << " B: " << n << '\n';
....
```

Ответ:
A: 3456 B:456

172. Чтобы прочитать символ из потока, не извлекая его, нужно использовать функцию ___?

Ответ:
peek();

173. Если из потока требуется ввести данные (raw data), которые включают и 0, следует использовать функцию ___?

Ответ:
istream& read (char* s, streamsize n);

174. Что выведет на экран приведенная программа, если пользователь ввел число 15?

Enter number:

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

void main(void)
{
    int p;
    cout<<"Enter number:"<<endl;
    cin>>p;
    cout<<p<<" : "<<hex<<p<<" : "<<oct<<p<<<'\n'
    cout<<endl;
}
```

Ответ:
15 : f : 17

175. При установке флага форматирования

cout.setf(ios::showpoint);

и выводе на экран значения вещественного числа 76.500000 ___.

Ответ:
будут показаны все завершающие 0 и на экране показано 76.500000

176. Что будет результатом выполнения следующей программы?

1234567890

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

void main(void)
{
    long p = 30000;
    cout<<"\n1234567890\n";

    cout.setf(ios::right,ios::adjustfield);
    cout<<setw(10)<<p<<endl;
}
```

Ответ:
30000

177. Что будет результатом выполнения следующей программы?

1234567890

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

void main(void)
{
    long p = 30000;
    cout<<"\n1234567890\n";

    cout.setf(ios::left,ios::adjustfield);
    cout<<setw(10)<<p<<endl;
}
```

Ответ:
30000

178. Что будет результатом выполнения следующей программы?

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

void main(void)
{
    long p = 30000;

    cout.setf(ios::right,ios::adjustfield);
    cout.fill('#');
    cout<<setw(10)<<dec<<p<<" | ";

    cout.setf(ios::left,ios::adjustfield);
    cout<<setw(10)<<setfill('$')<<p<<" | ";

    cout.setf(ios::internal,ios::adjustfield);
    cout<<setw(10)<<setfill('*')<<p<<endl;
}
```

Ответ:
#####30000 | 30000$$$$$ | *****30000

179. Что будет результатом выполнения следующих строк кода?

```
cout<<"123456789012345"<<endl;
cout.width(5);
cout<<111<<setw(10)<<555<<endl;
```

Ответ:
123456789012345
  111   555

180. Что будет результатом выполнения следующих строк кода?

```
cout<<1.11<<endl;
cout.setf(ios::showpoint);
cout<<setprecision(8)<<1.11<<endl;
```

Ответ:
1.11000
1.1100000

181. К какому классу принадлежат объекты файлы, если объекты этого класса ассоциируются с файлами открытыми для чтения?

Ответ:
ifstream

182. К какому классу принадлежат объекты файлы, если объекты этого класса ассоциируются с файлами открытыми для записи?

Ответ:
ofsteram

183. К какому классу принадлежат объекты файлы, если объекты этого класса ассоциируются с файлами открытыми для записи и чтения?

Ответ:
fstream

184. Чтобы связать объект поток с физическим файлом используется метод класса void open (const char* filename, ios_base::openmode mode = ios_base::in); поле mode определяет режим открытия файла, укажите битовый оператор, с помощью которого можно комбинировать режимы открытия файла?

Ответ:
|

185. Требуется открыть текстовый файл на ввод в режиме дополнения, какие из флагов следует выбрать?

Ответ:
ios::in

ios::app

186. Требуется открыть двоичный файл на вывод в режиме перезаписи, какие из флагов следует выбрать?

Ответ:
ios::binary

ios::trunk

ios::out

187. Выберете строки для того, чтобы открыть двоичный файл-поток на ввод в той же самой папке, что и исполняемая программа в режиме чтения. По завершении закрыть файл.

Ответ:
infile.open("tstfile.dat ", ios::in | ios::binary);

ifstream infile;

inpfile.close();

188. Выберете строки для того, чтобы открыть текстовый файл-поток на ввод в корневом каталоге диска С. По завершении закрыть файл.

Ответ:
```
ifstream  infile;

infile.open("c:\\tstfile.dat ");

inpfile.close();
```

189. Выберете строки для того, чтобы открыть текстовый файл-поток на ввод в той же самой папке, что и исполняемая программа, установить указатель в конец файла. По завершении закрыть файл.

Ответ:
```
infile.open("tstfile.dat ", ios::in | ios::ate);

inpfile.close();

ifstream  infile;
```

190. Выберете строки для того, чтобы открыть текстовый файл-поток на вывод в той же самой папке, что и исполняемая программа, в режиме дополнения. По завершении закрыть файл.

Ответ:
```
ofstream  outfile;

outfile.open("tstfile.dat", ios::out | ios::app);

outfile.close();
```

191. Выберете строки для того, чтобы открыть существующий текстовый файл-поток на вывод в той же самой папке, что и исполняемая программа. По завершении закрыть файл.

Ответ:
```
ofstream  outfile;

outfile.open("tstfile.dat", ios::out | ios::nocreate);

outfile.close();
```

192. Выберете строки для того, чтобы открыть текстовый файл-поток на вывод в корневой папке диска С. По завершении закрыть файл.

Ответ:
```
ofstream  outfile;

outfile.open("c:\\tstfile.dat ");

outfile.close();
```

193. Связать объект файл-поток с физическим файлом можно при помощи метода open, либо при помощи конструктора потока с параметрами.

Ответ:
Истина

194. При работе с потоком-файлом требуется выполнять проверку ошибок, для того что бы быть уверенным в успешном завершении намеченных действий, в противном случае могут возникнуть сообщения об ошибках либо вызваны обработчики ошибок.

Ответ:
Истина

195. Выберете комбинацию бит, если состояние потока - норма.
goodbit eofbit failbit badbit

Ответ:
true false false false

196. Выберете комбинацию бит, если состояние потока - считаны все символы.
goodbit eofbit failbit badbit

Ответ:
false true false false

197. Выберете комбинацию бит, если состояние потока – ошибки логики работы потока, возможно возобновление работы потока.
goodbit eofbit failbit badbit

Ответ:
false false true false

198. Выберете комбинацию бит, если состояние потока - потеря целостности потока, что приводит к невозможности нормального выполнения операций с потоком.
goodbit eofbit failbit badbit

Ответ:
false false true true

199. Какая из приведенных функций определения состояния потока вернет true если установлен бит goodbit.

Ответ:
bool good() const;

200. Какая из приведенных функций определения состояния потока вернет true если установлен бит eofbit.

Ответ:
bool eof() const;

201. Какая из приведенных функций определения состояния потока вернет true если установлен бит failbit и/или baddit.

Ответ:
bool fail() const;

202. Какая из приведенных функций определения состояния потока вернет true если установлен бит badbit.

Ответ:
bool bad() const;

203. Какие из приведенных функций позволяют выполнять операции в потоке файле в режиме произвольного доступа (Random File Processing) если файл открыт на чтение.

Ответ:
istream&seekg(streampos pos);

istream&seekg(streamoff off, ios_base::seekdir dir);

204. Какие из приведенных функций позволяют выполнять операции в потоке файле в режиме произвольного доступа (RandomFileProcessing) если файл открыт на запись.

Ответ:
ostream&seekp(streampospos);

ostream&seekp(streamoff off, ios_base::seekdir dir);

205. Если требуется извлечь символы из потока не считывая их из потока, какую из функций следует использовать?

Ответ:
istream& ignore (streamsize n = 1, int delim = EOF);

206. Что будет результатом выполнения следующей программы? Если файл содержит следующую строку:
0123456789A0A1A2A3A4A5A6A7A8A9

```
#include <iostream>     // std::cout
#include <fstream>      // std::ifstream
using namespace std;

int main () {

  std::ifstream is ("test.txt", std::ifstream::binary);
  if (is) {
    is.seekg (0, is.end);
    int temp = is.tellg();
    is.seekg (0, is.beg);

    std::cout << "Result = " << temp << "\n";
  }
  return 0;
}
```

Ответ:
Result = 30

207. Что будет результатом выполнения следующей программы? Если файл содержит следующую строку:
0123456789A0A1A2A3A4A5A6A7A8A9

```
#include <iostream>     // std::cout
#include <fstream>      // std::ifstream
using namespace std;

int main () {

  std::ifstream is ("test.txt", std::ifstream::binary);
  if (is) {
    is.seekg (0, is.end);
    int length = is.tellg();
    char * buffer = new char [length];

    is.seekg (0, is.beg);
    is.seekg(-10, ios_base::end);
    is.read (buffer,length);
    length = is.gcount();
    for(int i = 0; i<length; i++)
        std::cout << (char)buffer[i];
    std::cout << std::endl;

    delete[] buffer;
  }
  return 0;
}
```

Ответ:
A5A6A7A8A9

208. Что будет результатом выполнения следующей программы? Если файл содержит следующую строку:
0123456789A0A1A2A3A4A5A6A7A8A9

```
#include <iostream>      // std::cout
#include <fstream>       // std::ifstream
using namespace std;

int main () {

  std::ifstream is ("test.txt", std::ifstream::binary);
  if (is) {
    is.seekg (0, is.end);
    int length = is.tellg();
    char * buffer = new char [length];

    is.seekg(10, is.beg);
    is.seekg(10, ios_base::cur);
    is.read (buffer,length);
    length = is.gcount();
    for(int i = 0; i<length; i++)
        std::cout << (char)buffer[i];
    std::cout << std::endl;

    delete[] buffer;
  }
  return 0;
}
```

Ответ:
A5A6A7A8A9

209. Что будет результатом выполнения следующей программы? Если файл содержит следующую строку:
0123456789A0A1A2A3A4A5A6A7A8A9

```
#include <iostream>      // std::cout
#include <fstream>       // std::ifstream
using namespace std;

int main () {

  std::ifstream is ("test.txt", std::ifstream::binary);
  if (is) {
    is.seekg (0, is.end);
    int length = is.tellg();
    char * buffer = new char [length];

    is.seekg(10, ios_base::beg);
    is.read (buffer,length);
    length = is.gcount();
    for(int i = 0; i<length; i++)
        std::cout << (char)buffer[i];
    std::cout << std::endl;

    delete[] buffer;
  }
  return 0;
}
```

Ответ:
A0A1A2A3A4A5A6A7A8A9

210. Тип паттерна Chain of responsibility (Цепочка обязанностей)

Ответ:
поведенческий

211. Тип паттерна Command (Команда)

Ответ:
поведенческий

212. Тип паттерна Observer (Наблюдатель)

Ответ:
поведенческий

213. Тип паттерна State (Состояние)

Ответ:
поведенческий

214. Тип паттерна Memento (Хранитель)

Ответ:
поведенческий

215. Тип паттерна Interpreter (Интерпретатор)

Ответ:
поведенческий

216. Тип паттерна Strategy (Стратегия)

Ответ:
поведенческий

217. Тип паттерна Iterator (Итератор)

Ответ:
поведенческий

218. Тип паттерна Template Method (Шаблонный метод)

Ответ:
поведенческий

219. Тип паттерна Mediator (Посредник)

Ответ:
поведенческий

220. Тип паттерна Visitor (Посетитель)

Ответ:
поведенческий

221. Тип паттерна Adapter (Адаптер)

Ответ:
структурный

222. Тип паттерна Bridge (Мост)

Ответ:
структурный

223. Тип паттерна Composite (Компоновщик)

Ответ:
структурный

224. Тип паттерна Decorator (Декоратор)

Ответ:
структурный

225. Тип паттерна Facade (Фасад)

Ответ:
структурный

226. Тип паттерна Fliweight (Приспособленец)

Ответ:
структурный

227. Тип паттерна Proxy (Прокси)

Ответ:
структурный

228. Тип паттерна Abstract Factory (Абстрактная фабрика)

Ответ:
порождающий

229. Тип паттерна Builder (Строитель)

Ответ:
порождающий

230. Тип паттерна Factory Method (Фабричный метод)

Ответ:
порождающий

231. Тип паттерна Prototype (Прототип)

Ответ:
порождающий

232. Тип паттерна Singleton (Одиночка)

Ответ:
порождающий

233. К порождающим паттернам относятся

Ответ:
Singleton (Одиночка)

Prototype (Прототип)

234. К порождающим паттернам не относятся

Ответ:
Fliweight (Приспособленец)

Template Method (Шаблонный метод)

Adapter (Адаптер)

235. К порождающим паттернам относятся

Ответ:
Factory Method (Фабричный метод)

Builder (Строитель)

236. К порождающим паттернам не относятся

Ответ:
Composite (Компоновщик)

Facade (Фасад)

237. К структурным паттернам относятся

Ответ:
Decorator (Декоратор)

Proxy (Прокси)

238. К структурным паттернам не относятся

Ответ:
Interpreter (Интерпретатор)

Singleton (Одиночка)

239. К структурным паттернам относятся

Ответ:
Composite (Компоновщик)

Fliweight (Приспособленец)

240. К структурным паттернам не относятся

Ответ:
Mediator (Посредник)

Interpreter (Интерпретатор)

Singleton (Одиночка)

241. К поведенческим паттернам относятся

Ответ:
Interpreter (Интерпретатор)

Iterator (Итератор)

242. К поведенческим паттернам не относятся

Ответ:
Factory Method (Фабричный метод)

Abstract Factory (Абстрактная фабрика)

243. К поведенческим паттернам относятся

Ответ:
Strategy (Стратегия)

Mediator (Посредник)

Memento (Хранитель)

244. К поведенческим паттернам не относятся

Ответ:
Proxy (Прокси)

Fliweight (Приспособленец)

245. Установите соответствие между паттерном и его типом

Ответ:
Chain of responsibility (Цепочка обязанностей)
==========
поведенческий

Fliweight (Приспособленец)
==========
структурный

Factory Method (Фабричный метод)
==========
порождающий

246. Установите соответствие между паттерном и его типом

Ответ:
Template Method (Шаблонный метод)
==========
поведенческий

Bridge (Мост)
==========
структурный

Singleton (Одиночка)
==========
порождающий

247. Установите соответствие между паттерном и его типом

Ответ:
Command (Команда)
==========

поведенческий

Composite (Компоновщик)
==========
структурный

Prototype (Прототип)
==========
порождающий

248. Установите соответствие между паттерном и его типом

Ответ:
Observer (Наблюдатель)
==========
поведенческий

Decorator (Декоратор)
==========
структурный

Abstract Factory (Абстрактная фабрика)
==========
порождающий

249. Установите соответствие между паттерном и его типом

Ответ:
State (Состояние)
==========
поведенческий

Facade (Фасад)
==========
структурный

Factory Method (Фабричный метод)
==========
порождающий

250. Установите соответствие между паттерном и его типом

Ответ:
Mediator (Посредник)
==========

поведенческий

Bridge (Мост)
==========
структурный

Singleton (Одиночка)
==========
порождающий

251. Паттерн Chain of responsibility (Цепочка обязанностей)

Ответ:
Избегает связывания отправителя запроса с его получателем, давая возможность обработать запрос боле чем одному объекту

Связывает объекты-получатели и передает запрос по цепочке пока объект не обработает его

252. Паттерн Command (Команда)

Ответ:
Инкапсулирует запрос в виде объекта, позволяя передавать их клиентам в качестве параметров, ставить в очередь, логировать, а также поддерживает отмену операций

253. Паттерн Memento (Хранитель)

Ответ:
Не нарушая инкапсуляцию, определяет и сохраняет внутреннее состояние объекта и позволяет позже восстановить объект в этом состоянии

254. Паттерн Observer (Наблюдатель)

Ответ:
Определяет зависимость «один ко многим» между объектами так, что когда один объект меняет свое состояние, все зависимые объекты оповещаются и обновляются автоматически

255. Паттерн State (Состояние)

Ответ:
Позволяет объекту изменить свое поведение в зависимости от внутреннего состояния

256. Паттерн Interpreter (Интерпретатор)

Ответ:
Получая формальный язык, определяет представление его грамматики и интерпретатор, использующий это представление для его обработки выражений языка

257. Паттерн Strategy (Стратегия)

Ответ:
Определяет группу алгоритмов, инкапсулирует их и делает взаимозаменяемыми. Позволяет изменять алгоритм независимо от клиентов, его использующих

258. Паттерна Iterator (Итератор)

Ответ:
Предоставляет способ последовательного доступа к элементам множества, независимо от его внутреннего устройства

259. Паттерн Template Method (Шаблонный метод)

Ответ:
Определяет алгоритм, некоторые этапы которого делегируются подклассам. Позволяет подклассам переопределить эти этапы, не меняя структуру алгоритма

260. Паттерн Mediator (Посредник)

Ответ:
Определяет объект, инкапсулирующий способ взаимодействия объектов. Обеспечивает слабую связь, избавляя объекты от необходимости прямо ссылать друг на друга, дает возможность независимо изменять их взаимодействие

261. Паттерн Visitor (Посетитель)

Ответ:
Представляет собой операцию, которая будет выполнена над объектами группы классов

Дает возможность определить новую операцию без изменения кода классов, над которыми эта операция выполняется

262. Паттерн Adapter (Адаптер)

Ответ:
Конвертирует интерфейс класса в другой интерфейс, ожидаемый клиентом

Позволяет классам с разными интерфейсами работать вместе

263. Паттерн Bridge (Мост)

Ответ:
Разделяет абстракцию и реализацию так, чтобы они могли изменяться независимо

264. Паттерн Composite (Компоновщик)

Ответ:
Компонует объекты в древовидную структуру, представляя их в виде иерархии. Позволяет клиенту одинаково обращаться как к отдельному объекту, так и к целому дереву

265. Паттерн Decorator (Декоратор)

Ответ:
Динамически предоставляет объекту дополнительные возможности

Представляет собой гибкую альтернативу наследованию для расширения функциональности

266. Паттерн Facade (Фасад)

Ответ:
Предоставляет единый интерфейс к группе интерфейсов подсистемы

Определяет высокоуровневый интерфейс, делая подсистему проще для использования

267. Паттерн Fliweight (Приспособленец)

Ответ:
Благодаря совместному использованию, поддерживает эффективную работу с большим количеством объектов

268. Паттерн Proxy (Прокси)

Ответ:
Предоставляет замену другого объекта для контроля доступа к нему

269. Паттерн Abstract Factory (Абстрактная фабрика)

Ответ:
Предоставляет интерфейс для создания групп связанных или зависимых объектов, не указывая их конкретный класс

270. Паттерн Builder (Строитель)

Ответ:
Разделяет создание сложного объекта и инициализацию его состояния так, что одинаковый процесс построения может создать объекты с разным состоянием

271. Паттерн Factory Method (Фабричный метод)

Ответ:
Определяет интерфейс для создания объекта, но позволяет подклассам решать, какой класс инстанциировать

Позволяет делегировать создание объекта подклассам

272. Паттерн Prototype (Прототип)

Ответ:
Определяет несколько видов объектов, чтобы при создании использовать объект-прототип и создает новые объекты, копируя прототип

273. Паттерн Singleton (Одиночка)

Ответ:
Гарантирует, что класс имеет только один экземпляр и предоставляет глобальную точку доступа к нему

274. Избегает связывания отправителя запроса с его получателем, давая возможность обработать запрос боле чем одному объекту

Ответ:
Паттерн Chain of responsibility (Цепочка обязанностей)

275. Инкапсулирует запрос в виде объекта, позволяя передавать их клиентам в качестве параметров, ставить в очередь, логировать, а также поддерживает отмену операций

Ответ:
Паттерн Command (Команда)

276. Не нарушая инкапсуляцию, определяет и сохраняет внутреннее состояние объекта и позволяет позже восстановить объект в этом состоянии

Ответ:
Паттерн Memento (Хранитель)

277. Определяет зависимость «один ко многим» между объектами так, что когда один объект меняет свое состояние, все зависимые объекты оповещаются и обновляются автоматически

Ответ:
Паттерн Observer (Наблюдатель)

278. Позволяет объекту изменить свое поведение в зависимости от внутреннего состояния

Ответ:
Паттерн State (Состояние)

279. Получая формальный язык, определяет представление его грамматики и интерпретатор, использующий это представление для его обработки выражений языка

Ответ:
Паттерн Interpreter (Интерпретатор)

280. Определяет группу алгоритмов, инкапсулирует их и делает взаимозаменяемыми. Позволяет изменять алгоритм независимо от клиентов, его использующих

Ответ:
Паттерн Strategy (Стратегия)

281. Предоставляет способ последовательного доступа к элементам множества, независимо от его внутреннего устройства

Ответ:
Паттерна Iterator (Итератор)

282. Определяет алгоритм, некоторые этапы которого делегируются подклассам. Позволяет подклассам переопределить эти этапы, не меняя структуру алгоритма

Ответ:
Паттерн Template Method (Шаблонный метод)

283. Определяет объект, инкапсулирующий способ взаимодействия объектов. Обеспечивает слабую связь, избавляя объекты от необходимости прямо ссылать друг на друга, дает возможность независимо изменять их взаимодействие

Ответ:
Паттерн Mediator (Посредник)

284. Дает возможность определить новую операцию без изменения кода классов, над которыми эта операция выполняется

Ответ:
Паттерн Visitor (Посетитель)

285. Конвертирует интерфейс класса в другой интерфейс, ожидаемый клиентом.

Ответ:
Паттерн Adapter (Адаптер)

286. Компонует объекты в древовидную структуру, представляя их в виде иерархии. Позволяет клиенту одинаково обращаться как к отдельному объекту, так и к целому дереву.

Ответ:
Паттерн Bridge (Мост)

287. Компонует объекты в древовидную структуру, представляя их в виде иерархии. Позволяет клиенту одинаково обращаться как к отдельному объекту, так и к целому дереву

Ответ:
Паттерн Composite (Компоновщик)

288. Динамически предоставляет объекту дополнительные возможности. Представляет собой гибкую альтернативу наследованию для расширения функциональности

Ответ:
Паттерн Decorator (Декоратор)

289. Определяет высокоуровневый интерфейс, делая подсистему проще для использования

Ответ:
Паттерн Facade (Фасад)

290. Благодаря совместному использованию, поддерживает эффективную работу с большим количеством объектов

Ответ:
Паттерн Fliweight (Приспособленец)

291. Предоставляет замену другого объекта для контроля доступа к нему

Ответ:
Паттерн Proxy (Прокси)

292. Предоставляет интерфейс для создания групп связанных или зависимых объектов, не указывая их конкретный класс

Ответ:
Паттерн Abstract Factory (Абстрактная фабрика)

293. Разделяет создание сложного объекта и инициализацию его состояния так, что одинаковый процесс построения может создать объекты с разным состоянием

Ответ:
Паттерн Builder (Строитель)

294. Определяет интерфейс для создания объекта, но позволяет подклассам решать, какой класс инстанциировать

Ответ:
Паттерн Factory Method (Фабричный метод)

295. Определяет несколько видов объектов, чтобы при создании использовать объект-прототип и создает новые объекты, копируя прототип

Ответ:
Паттерн Prototype (Прототип)

296. Гарантирует, что класс имеет только один экземпляр и предоставляет глобальную точку доступа к нему

Ответ:
Паттерн Singleton (Одиночка)

297. Из каких элементов состоит паттерн abstract factory?

Ответ:
Abstract factory

Abstract product

Concrete factory

298. Из каких элементов состоит паттерн builder

Ответ:
Builder interface

Director Interface

Concrete Builder class

299. Из каких элементов состоит паттерн builder

Ответ:
Creator

Product

300. Паттерн Factory Method создает объекты разных типов, позволяя системе оставаться независимой как от самого процесса создания, так и от типов создаваемых объектов

Ответ:
Верно

301. Паттерн Factory Method создает объекты только одного типа, что позволяет системе оставаться независимой от самого процесса создания объектов

Ответ:
Не верно

302. В случае классического варианта реализации паттерна Factory Method даже для порождения единственного объекта необходимо создавать соответствующую фабрику

Ответ:
Верно

303. В случае классического варианта реализации паттерна Factory Method для порождения только одного объекта нет необходимости создавать соответствующую фабрику

Ответ:
Не верно

304. Паттерн Abstract Factory скрывает сам процесс порождения объектов, а также делает систему независимой от типов создаваемых объектов, специфичных для различных семейств или групп.

Ответ:
Верно

305. Паттерн Abstract Factory позволяет быстро настраивать систему на нужное семейство создаваемых объектов

Ответ:
Используя паттерн Abstract Factory, трудно добавлять новые типы создаваемых продуктов или заменять существующие, так как интерфейс базового класса абстрактной фабрики фиксирован.

Верно

Верно

306. Используя паттерн Abstract Factory, можно легко добавлять новые типы создаваемых продуктов или заменять существующие, так как интерфейс базового класса абстрактной фабрики не фиксирован.

Ответ:
Не верно

307. К достоинствам паттерна Builder можно отнести возможность контролировать процесс создания сложного продукта.

Ответ:
Верно

308. К достоинствам паттерна Builder можно отнести возможность получения разных представлений некоторых данных.

Ответ:
Верно

309. К недостаткам паттерна Builder можно отнести отсутствие возможности контролировать процесс создания сложного продукта.

Ответ:
Не верно

310. К недостаткам паттерна Builder можно отнести отсутствие возможности получения разных представлений некоторых данных.

Ответ:
Не верно

311. Используя паттерн Prototype, для создания новых объектов клиенту необязательно знать их конкретные классы.

Ответ:
Верно

312. К достоинствам паттерна Prototype можно отнести возможность гибкого управления процессом создания новых объектов за счет возможности динамических добавления и удаления прототипов в реестр.

Ответ:
Верно

313. К недостаткам паттерна Prototype можно отнести отсутствие возможности гибкого управления процессом создания новых объектов.

Ответ:
Не верно

314. Паттерн Singleton легко адаптировать для создания нужного числа экземпляров.

Ответ:
Верно

315. При использовании нескольких взаимозависимых одиночек (паттерн Singleton) их реализация может резко усложниться

Ответ:
Верно

316. Какой паттерн необходимо применить, когда решение о типе класса принимается во время его инстанцирования

Ответ:
Factory Method

317. 317. Дана следующая ситуация: Вы хотите создать семейства связанных объектов, которые будут использоваться как синонимы для настройки вашего приложения. Какой паттерн GoF наиболее подходит в этой ситуации

Ответ:
Abstract Factory

318. 318. Паттерн так же известен как Виртуальный Конструктор (Virtual Constructor) и используется для предоставления интерфейса создания экземпляров некоторых классов. В момент создания наследники могут определять, какой класс создавать. Это паттерн

Ответ:
Factory Method

319. Вы хотите, чтобы все клиенты использующие класс А использовали один и тот же объект класса А. Что вы должны сделать, чтобы достичь этой цели?

Ответ:
применить Singleton pattern к классу А

320. Вы хотите добавить уже написанный класс из другого приложения в свое приложение. Все классы вашего приложения имеют один интерфейс, а новый класс имеет совершенно другой интерфейс, но обладает нужной функциональностью. Какой рефакторинг необходимо предпринять, чтобы использовать этот класс с минимальными изменениями в вашем приложении?

Ответ:
применить паттерн Adapter

321. В чем разница между паттернами Адаптер и Декоратор?

Ответ:
Адаптер не добавляет новой функциональности в адаптируемый класс, в то время как Декоратор расширяет функциональность объекта

Оба вносят уровень связности между классом реализации и классом, который его использует

322. Что лучше определяет использование паттерна Компоновщик (Composite Design Pattern)

Ответ:
Паттерн Компоновщик позволяет обращаться с отдельными объектами и их объединениями единообразно

323. Паттерн, который так же известен как Обертка (Wrapper). Используется для добавления функциональности динамически, когда наследование не представляется возможным, поскольку это решение статическое

Ответ:
Decorator

324. Большое количество клиентов удаленно вызывают методы разрозненных реализаций или интерфейсов вашей системы, что приводит к большой нагрузки сети. Какой паттерн позволяет улучшить производительность вашей системы

Ответ:
Facade Pattern

325. Этот паттерн позволяет минимизировать использования памяти, позволяя подобным объектам хранить как можно больше одинаковых для них данных в одном разделяемом объекте

Ответ:
Flyweight

326. Какой паттерн используется в следующем примере? Каждый объект для одной буквы слова содержит информацию о ее графическом представлении и позицию. Но для предотвращения избыточного хранения графических представлений одних и тех же букв целесообразно извлечь графические представления в один разделяемый объект и хранить только их позицию внутри каждого слова

Ответ:
Flyweight

327. Какой паттерн может быть использован для отложенной (ленивой) загрузки данных?

Ответ:
Proxy

328. Паттерн Proxy лучше всего использовать для:

Ответ:
Контроль доступа к удаленному объекту

Доставка ресурсоемких объектов

Сохранение ссылки на объект, необходимой для его вызова

329. Какой паттерн используется, когда больше чем один объект может обработать запрос и нет сильной зависимости между ними

Ответ:
Chain of Responsibility

330. Какой паттерн используется для инкапсуляции запроса в виде объекта для поддержки операции отмены, логгирования или поддержки функционала транзакций

Ответ:
Command Pattern

331. ВАРИАНТ 1 В модели сообщений «Издатель - Подписчик» подписчики подписываются на определенную тему и получают уведомления, когда в теме появляется новое сообщение
ВАРИАНТ 2 В модели сообщений «Издатель - Подписчик» подписчики регистрируются у поставщика и получают сообщения при каждом выполнении предварительно заданного события поставщик автоматически уведомляет всех подписчиков путем вызова одного из их методов
Какой паттерн наиболее точно определяется этой моделью?

Ответ:
Observer

332. Паттерн, который предназначен для определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости

Ответ:
Strategy Pattern

333. В каких случаях рекомендуется применять паттерн Strategy

Ответ:
Необходимо предоставить выбор из нескольких алгоритмов

Несколько классов имеют одинаковый интерфейс для использования и отличаются только реализацией (поведением)

334. Вы разрабатываете приложение для рекламной компании, которая производит разного рода издания, такие как книги, статьи, брошюры и т.д. Компания выпускает эти продукты в нескольких медиа-форматах: печатные материалы, CD, DVD, интернет-сайты и т.д. Какой паттерн вы примените для моделирования иерархии продуктов компании (Издания, Медиа-форматы)?

Ответ:
Паттерн Bridge для создания независимых иерархий Изданий и Медиа-форматов и разделения абстракции и реализации на две отдельные иерархии классов так, чтобы их можно было изменять независимо друг от друга

335. Какой паттерн используется для разделения абстракции и реализации на две отдельные иерархии классов так, чтобы их можно было изменять независимо друг от друга

Ответ:
Bridge design pattern

336. В каких случаях необходимо применять паттерн Abstract Factory

Ответ:
Для создания группы связанных объектов исключая возможность одновременного использования объектов из разных групп в одном контексте

Для создания системы, независимой как от процесса создания новых объектов, так и от типов порождаемых объектов

Для разделения создания объектов от их использования

337. Какие следствия применения паттерна Abstract Factory

Ответ:
Более легкое взаимодействие между объектами одной иерархии

Взаимозаменяемость иерархий объектов

338. Какой паттерн используется для процесса поэтапного конструирования сложного объекта, в результате которого могут получаться разные представления этого объекта

Ответ:
Builder design pattern

339. В каких ситуациях необходимо использовать паттерн Builder

Ответ:
Для абстрагирования шагов создания объектов, что обеспечивает поддержку различных реализаций

Для применения сходных процедур создания различных представлений объектов

340. Какой паттерн применяется для ограничение количества экземпляров класса до одного объекта

Ответ:
Singleton design pattern

341. Какие следствия применения паттерна Prototype

Ответ:
Каждый конкретный класс прототипа должен реализовать метод клонирования

Создание объектов, точные классы которых становятся известными уже на стадии выполнения программы

Уменьшается иерархия классов по сравнению с другими паттернами

342. В каких ситуациях необходимо использовать паттерн Prototype

Ответ:
определение классов, которые будут инстанциированы в момент выполнения

предотвращение разрастания иерархии классов

343. Какие следствия применения паттерна Factory Method

Ответ:
Разделение кода клиента от конкретных классов приложения

Гибкий механизм инстанциирования объектов по сравнению со стандартными конструкторами

344. В каких ситуациях необходимо использовать паттерн Factory Method

Ответ:
Управление созданием объектов на основе интерфейса

Возможность конкретным реализациям определять, какие подклассы будет инстанциированы

345. Какой паттерн необходимо использовать для создания объектов, позволяя при этом конкретным реализациям определять, какие подклассы будут инстанциированы

Ответ:
Factory method design pattern

346. В каких ситуациях необходимо использовать паттерн Builder

Ответ:
абстрагирование шагов создания сложных объектов

построения различных представлений сложных объектов на основе конкретных реализаций процедуры конструирования

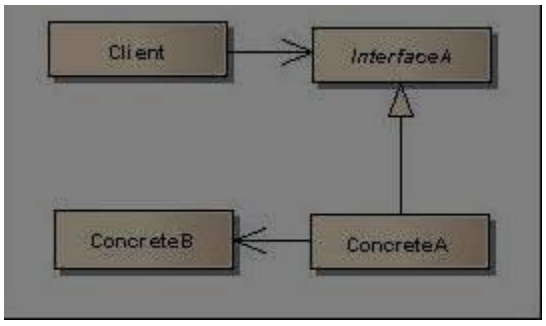
347. Какие следствия применения шаблона Builder

Ответ:
Более легкое добавление новой реализации объектов

Разделение конструирования объекта от его представления

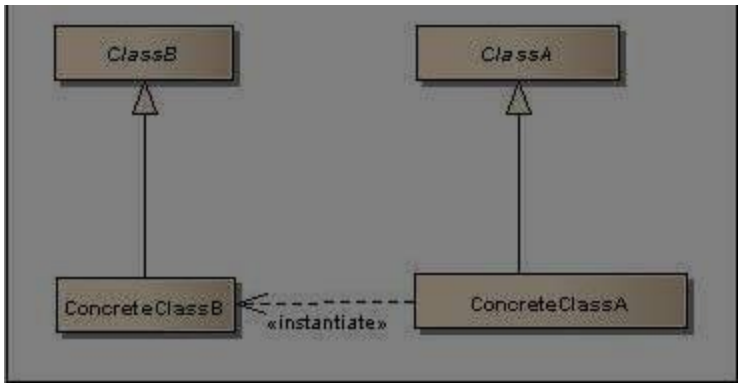
Более тонкий контроль над процедурами создания объектов

348. Какой из структурных паттернов изображен на следующей диаграмме?



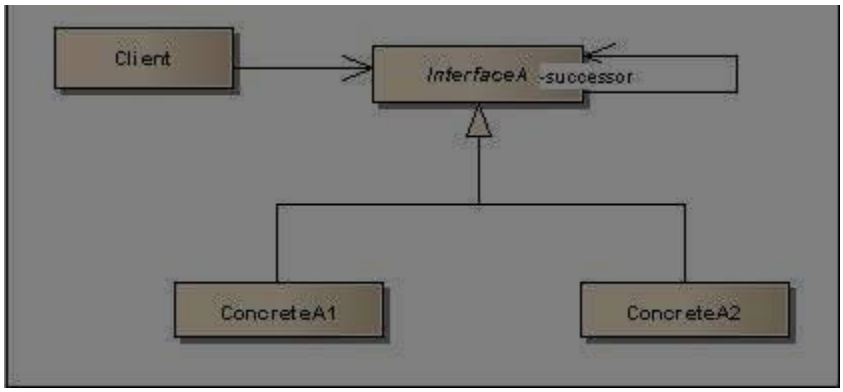
Ответ:
Adapter

349. Какой из порождающих паттернов изображен на следующей диаграмме?



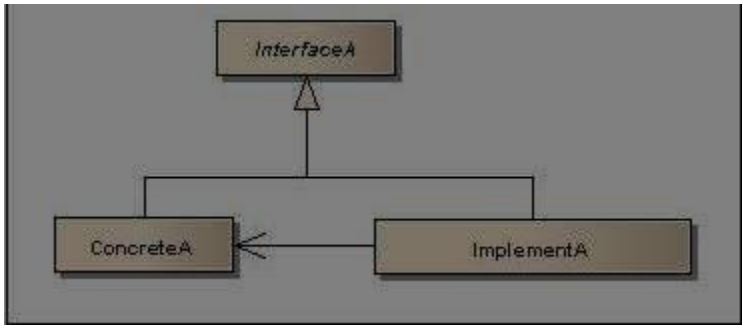
Ответ:
Factory Method

350. Какой из поведенческих паттернов изображен на следующей диаграмме?



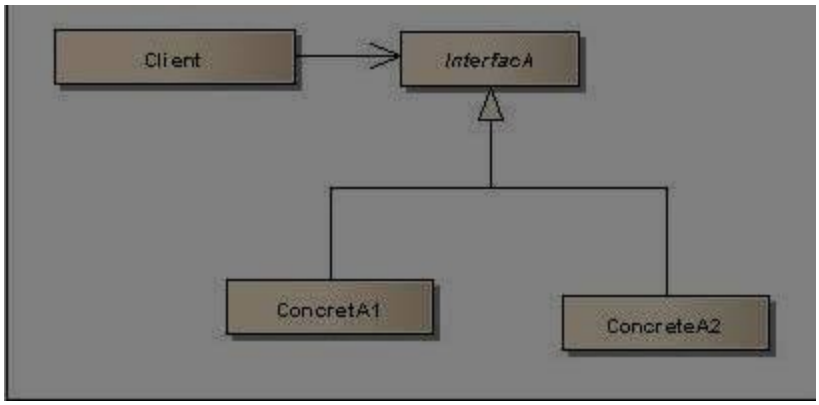
Ответ:
Chain Of Responsibility

351. Какой из структурных паттернов изображен на следующей диаграмме?



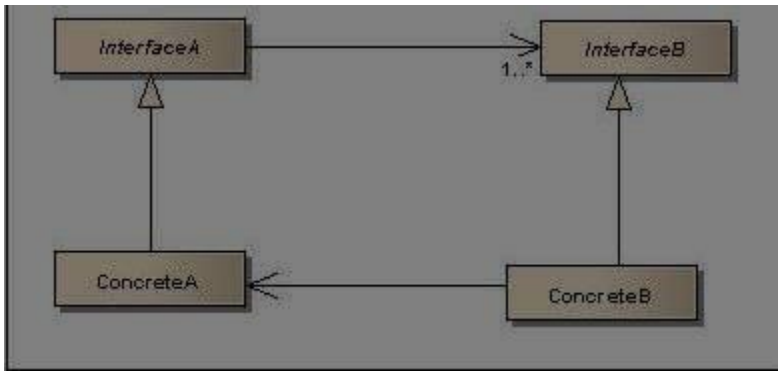
Ответ:
Proxy

352. Какой из порождающих паттернов изображен на следующей диаграмме?



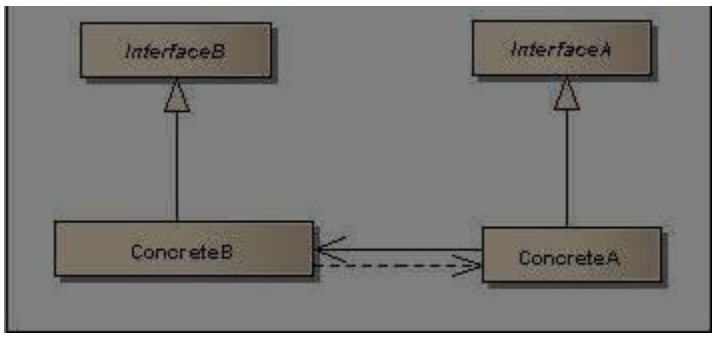
Ответ:
Prototype

353. Какой из поведенческих паттернов изображен на следующей диаграмме?



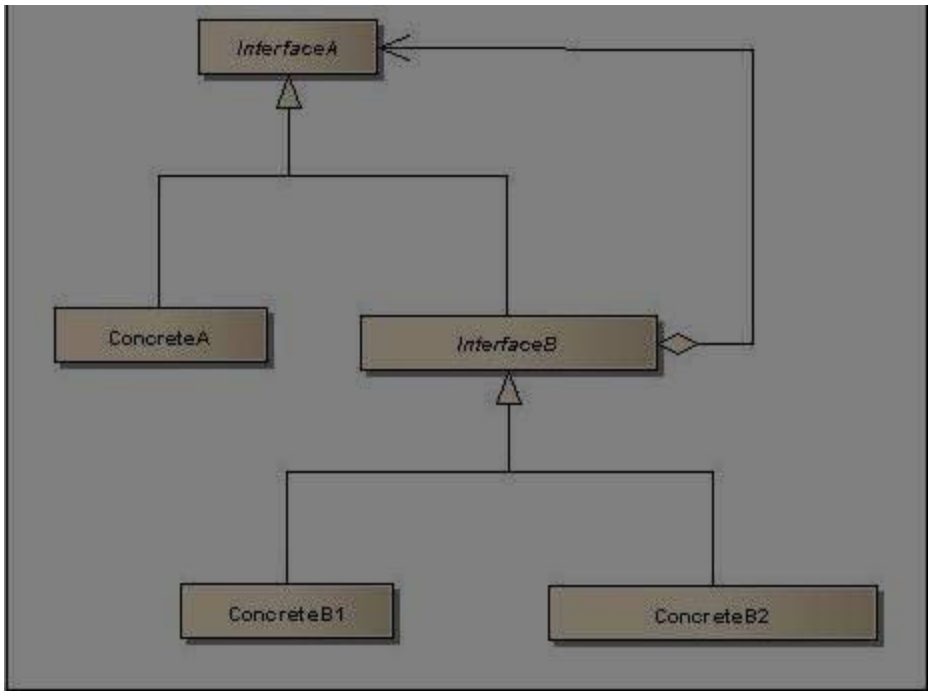
Ответ:
Observer

354. Какой из поведенческих паттернов изображен на следующей диаграмме?



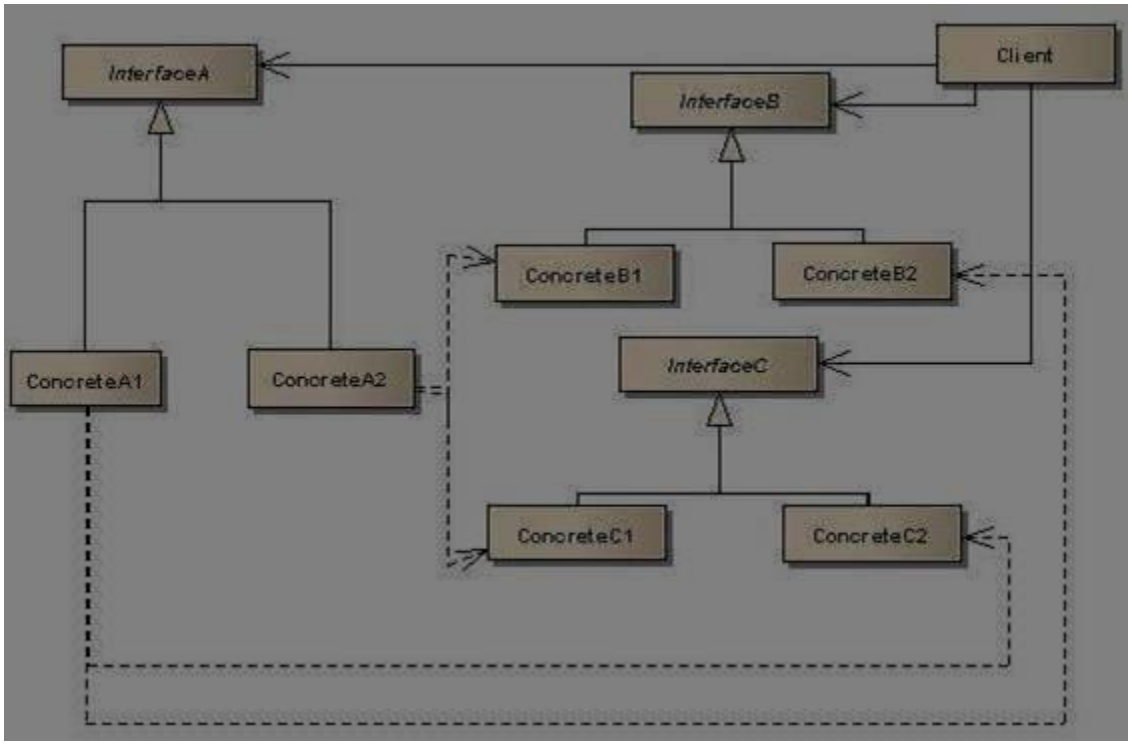
Ответ:
Iterator

355. Какой из структурных паттернов изображен на следующей диаграмме?



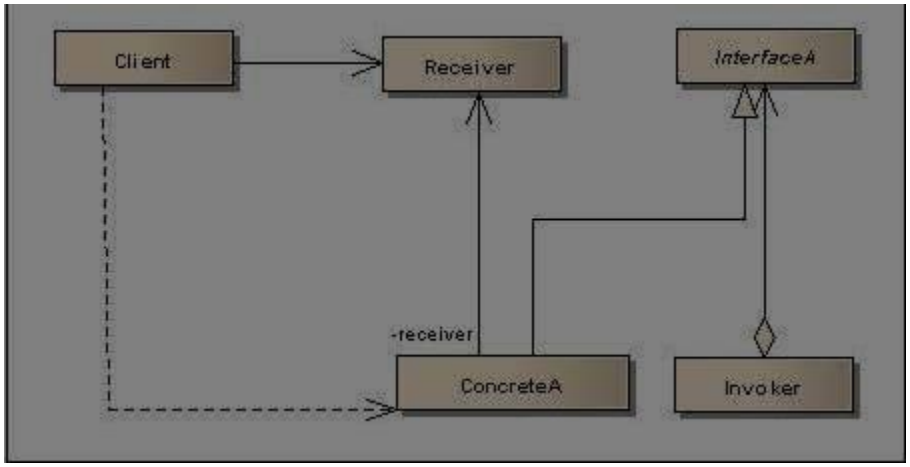
Ответ:
Decorator

356. Какой из порождающих паттернов изображен на следующей диаграмме?



Ответ:
Abstract Factory

357. Какой из поведенческих паттернов изображен на следующей диаграмме?



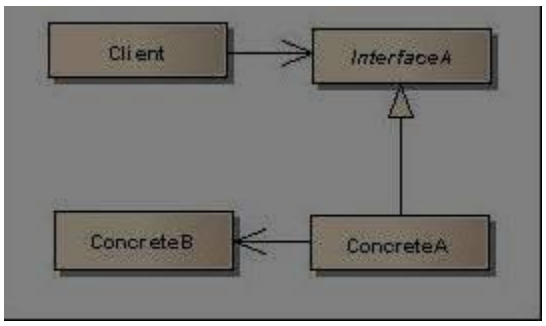
Ответ:
Command

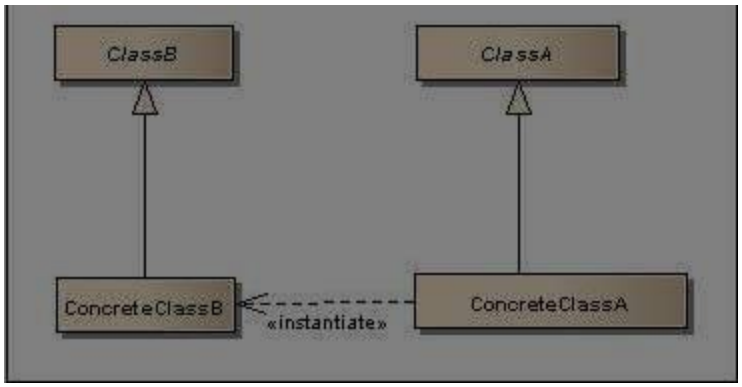
358. Установите соответствие между диаграммой и названием паттерна

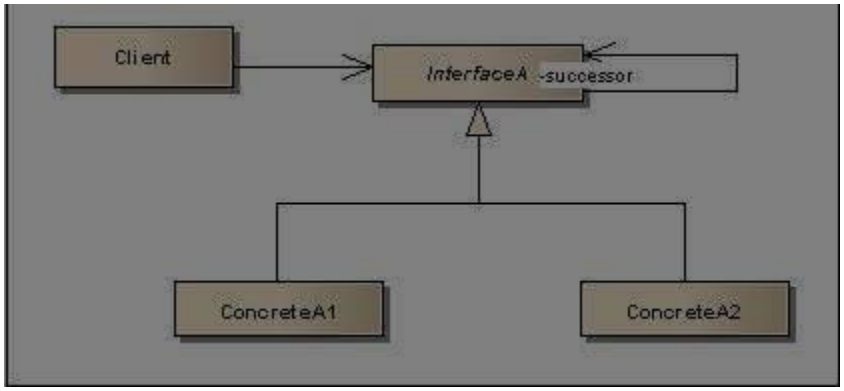
Ответ:
==========
Adapter

==========
Factory Method

==========
Chain Of Responsibility





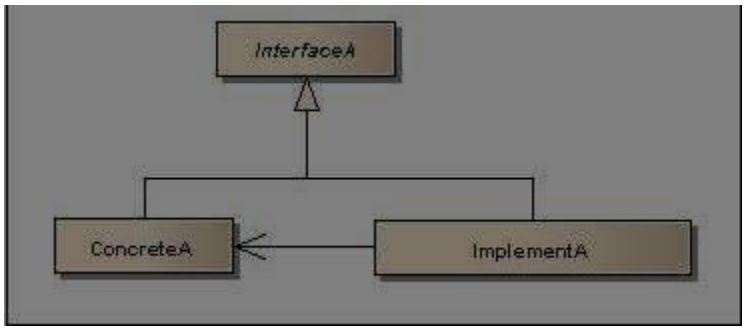


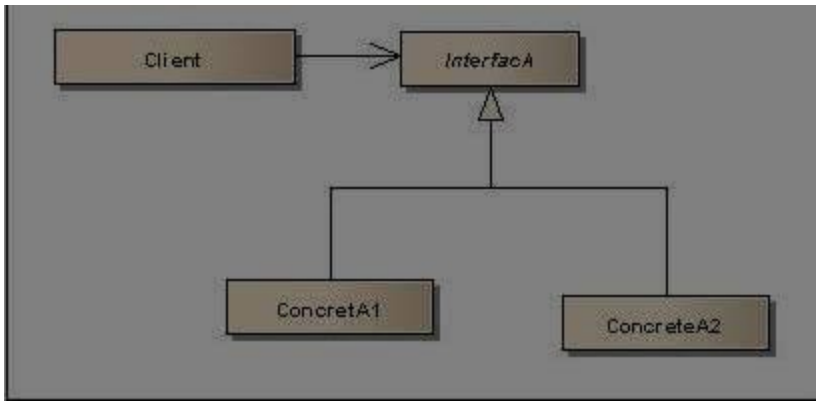
359. Установите соответствие между диаграммой и названием паттерна

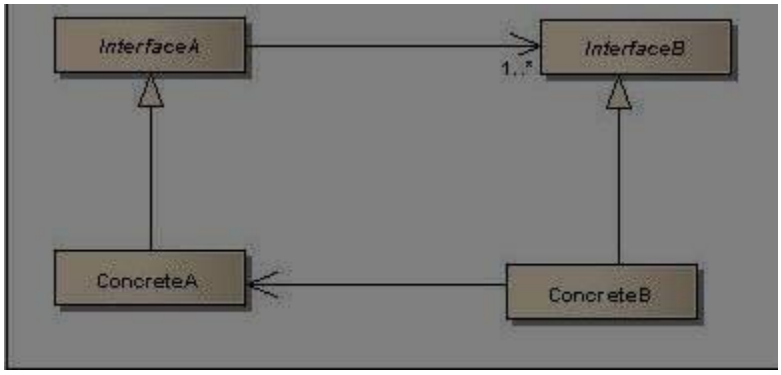
Ответ:
==========
Proxy

==========
Prototype

==========
Observer





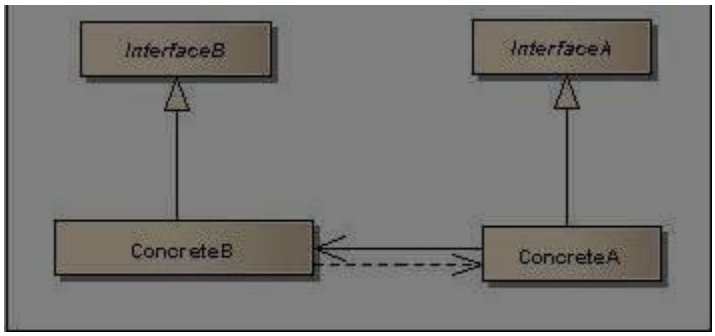


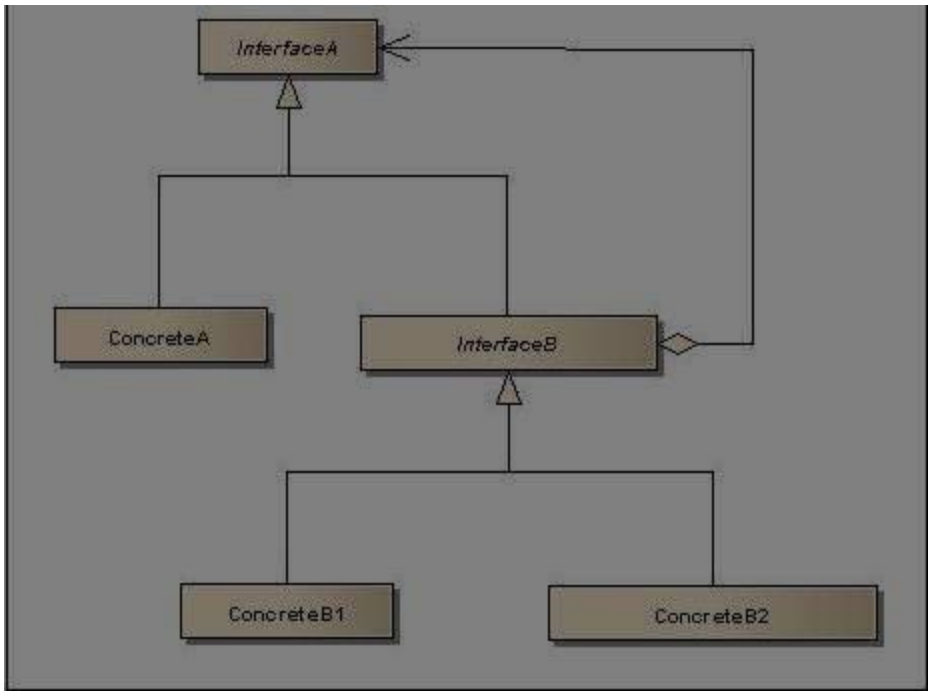
360. Установите соответствие между диаграммой и названием паттерна

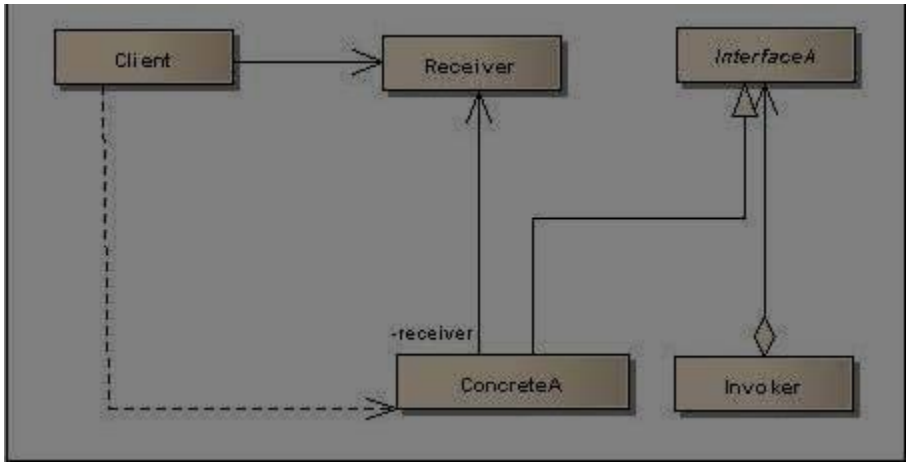
Ответ:
==========
Iterator

==========
Decorator

==========
Command







361. Установите соответствие между диаграммой и названием паттерна

Ответ:

==========

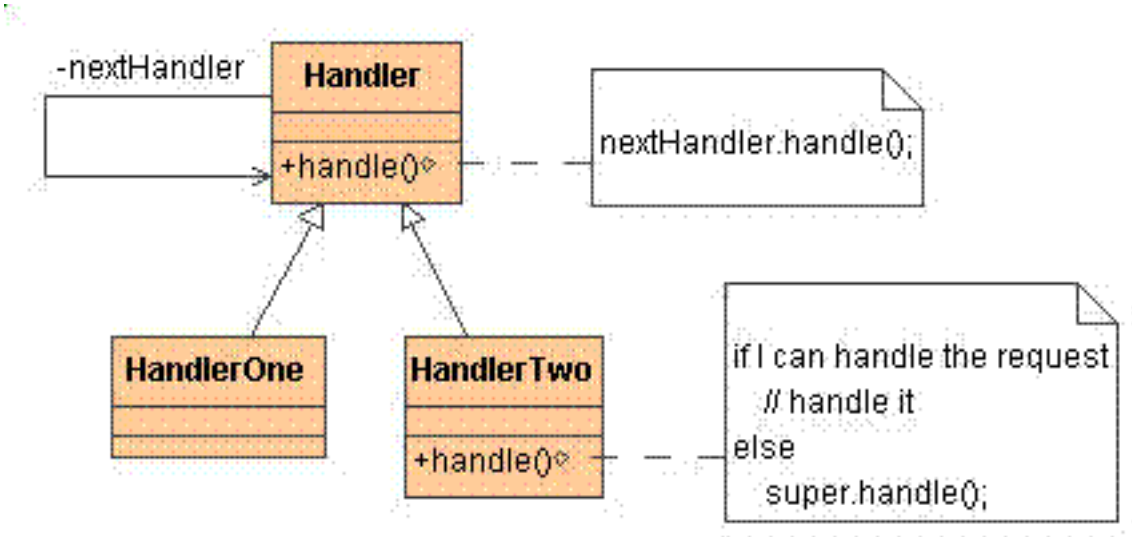
Паттерн Chain of responsibility (Цепочка обязанностей)

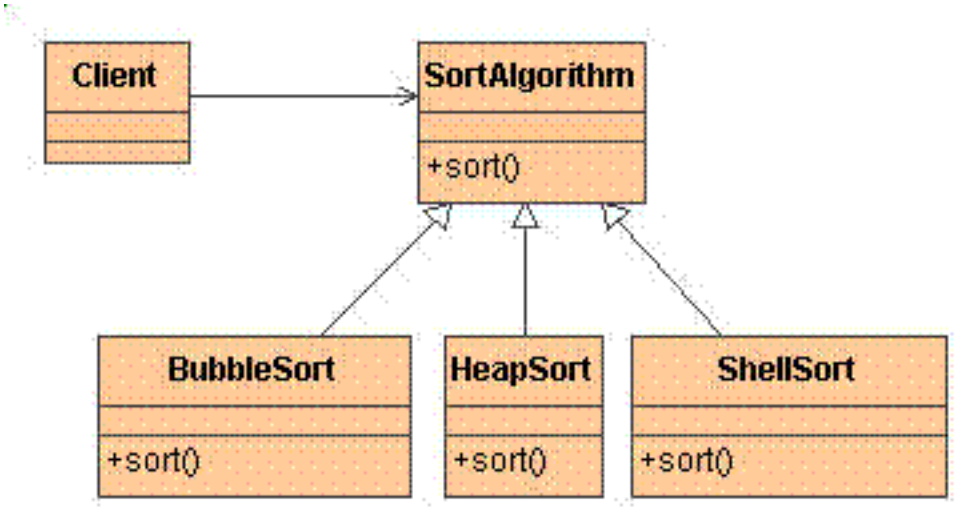
==========

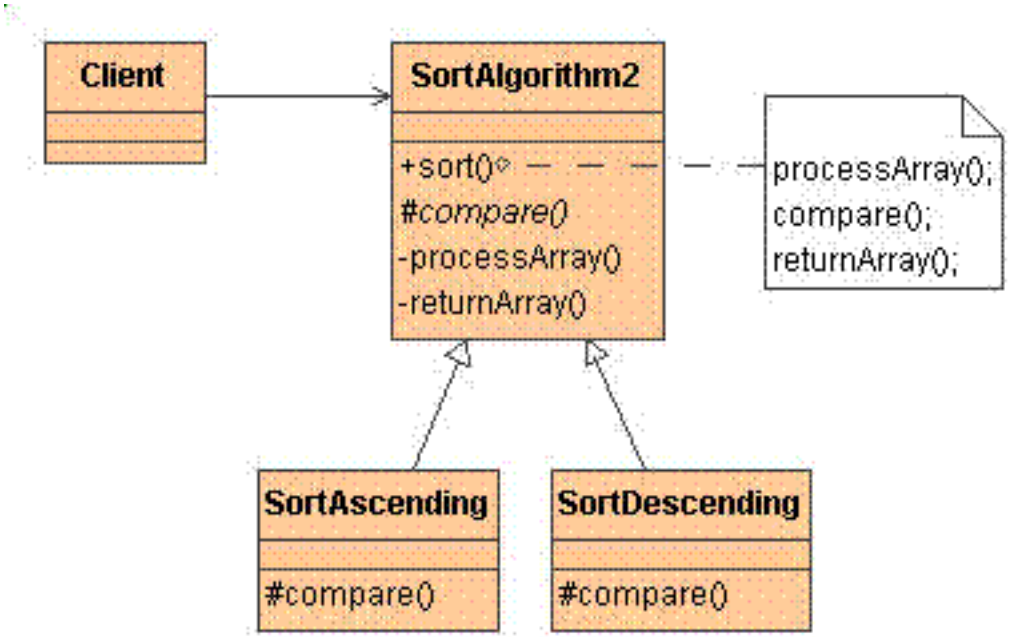
Паттерн Strategy (Стратегия)

==========

Паттерн Template Method (Шаблонный метод)


-nextHandler
Handler
+handle()
nextHandler.handle();
HandlerOne
HandlerTwo
+handle()
if I can handle the request
// handle it
else
super.handle();


Client
SortAlgorithm
+sort()
BubbleSort
+sort()
HeapSort
+sort()
ShellSort
+sort()

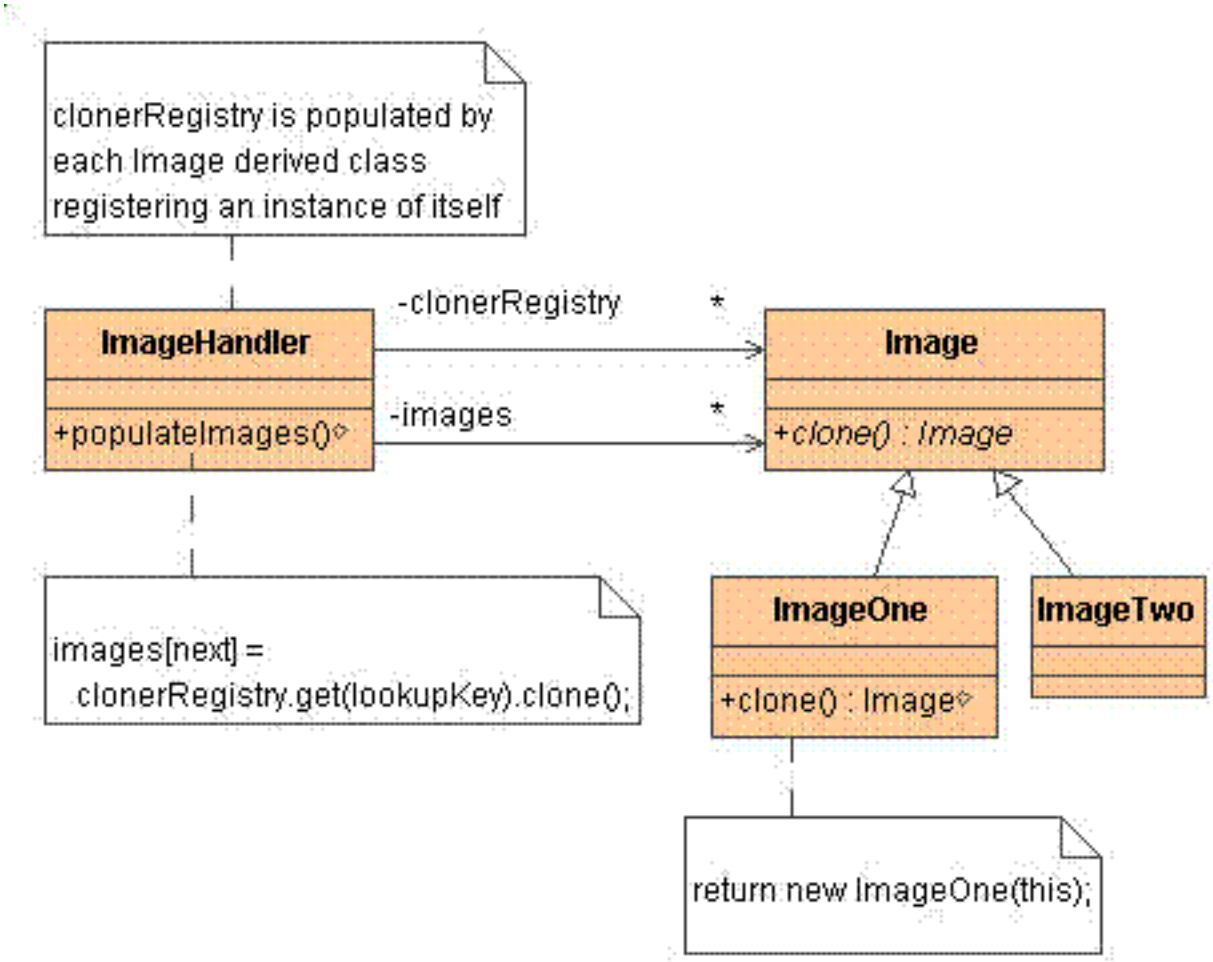


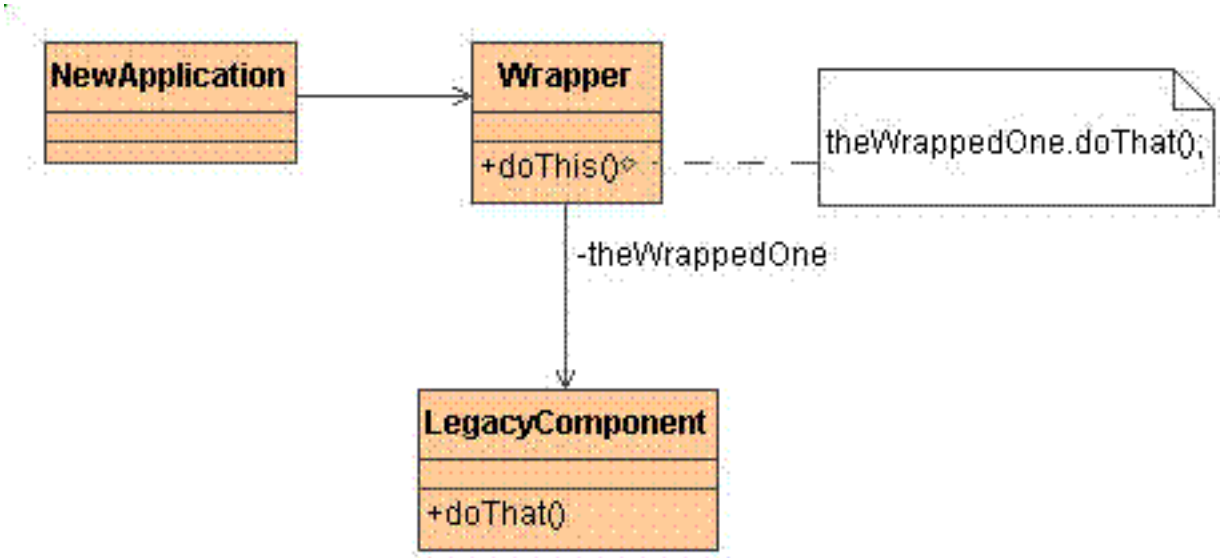
362. Установите соответствие между диаграммой и названием паттерна

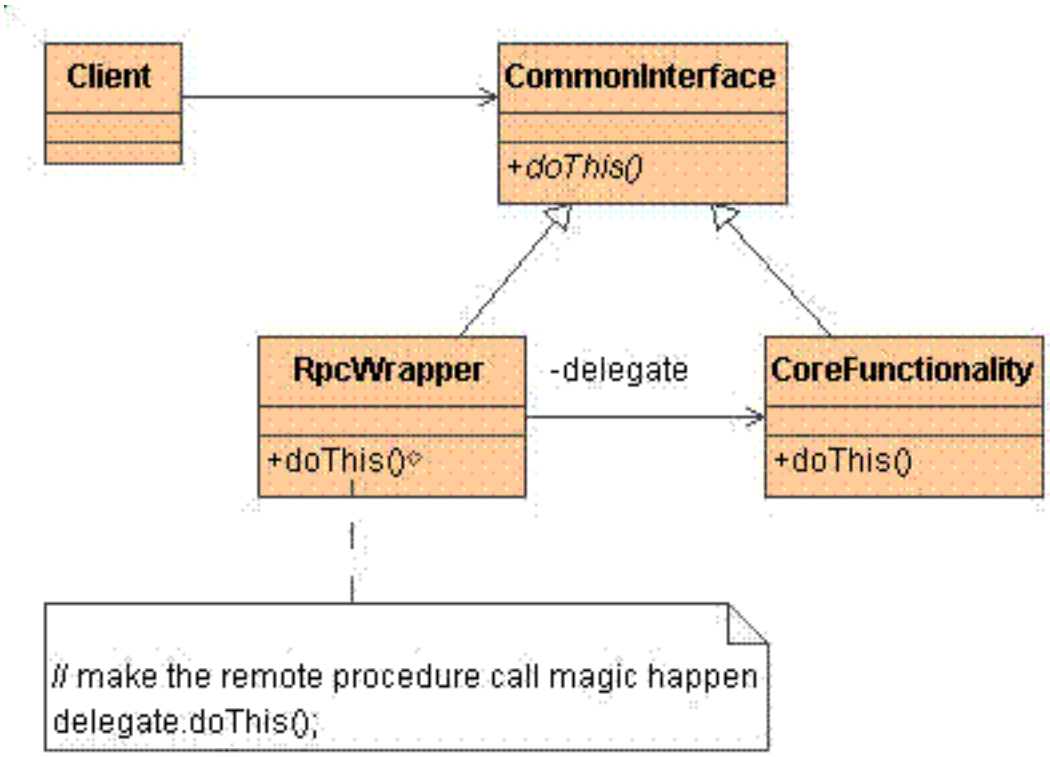
Ответ:
==========
Паттерн Prototype (Прототип)

==========
Паттерн Adapter (Адаптер)

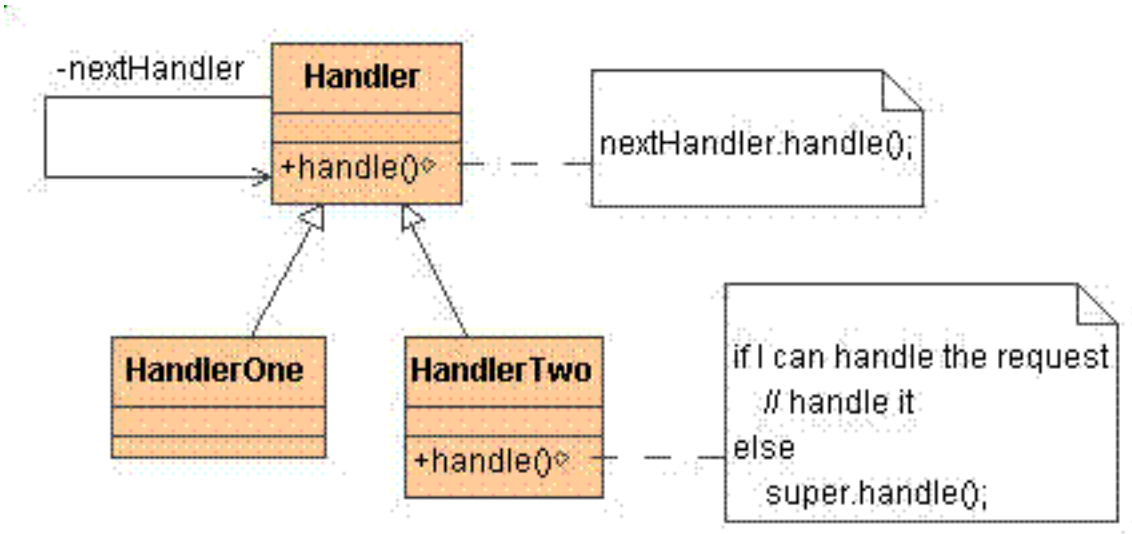
==========
Паттерн Proxy (Прокси)


clonerRegistry is populated by each Image derived class registering an instance of itself
ImageHandler
+populateImages()
-clonerRegistry
*
-images
*
Image
+clone() : Image
images[next] = clonerRegistry.get(lookupKey).clone();
ImageOne
+clone() : Image
ImageTwo
return new ImageOne(this);


NewApplication
Wrapper
+doThis()
theWrappedOne.doThat();
-theWrappedOne
LegacyComponent
+doThat()

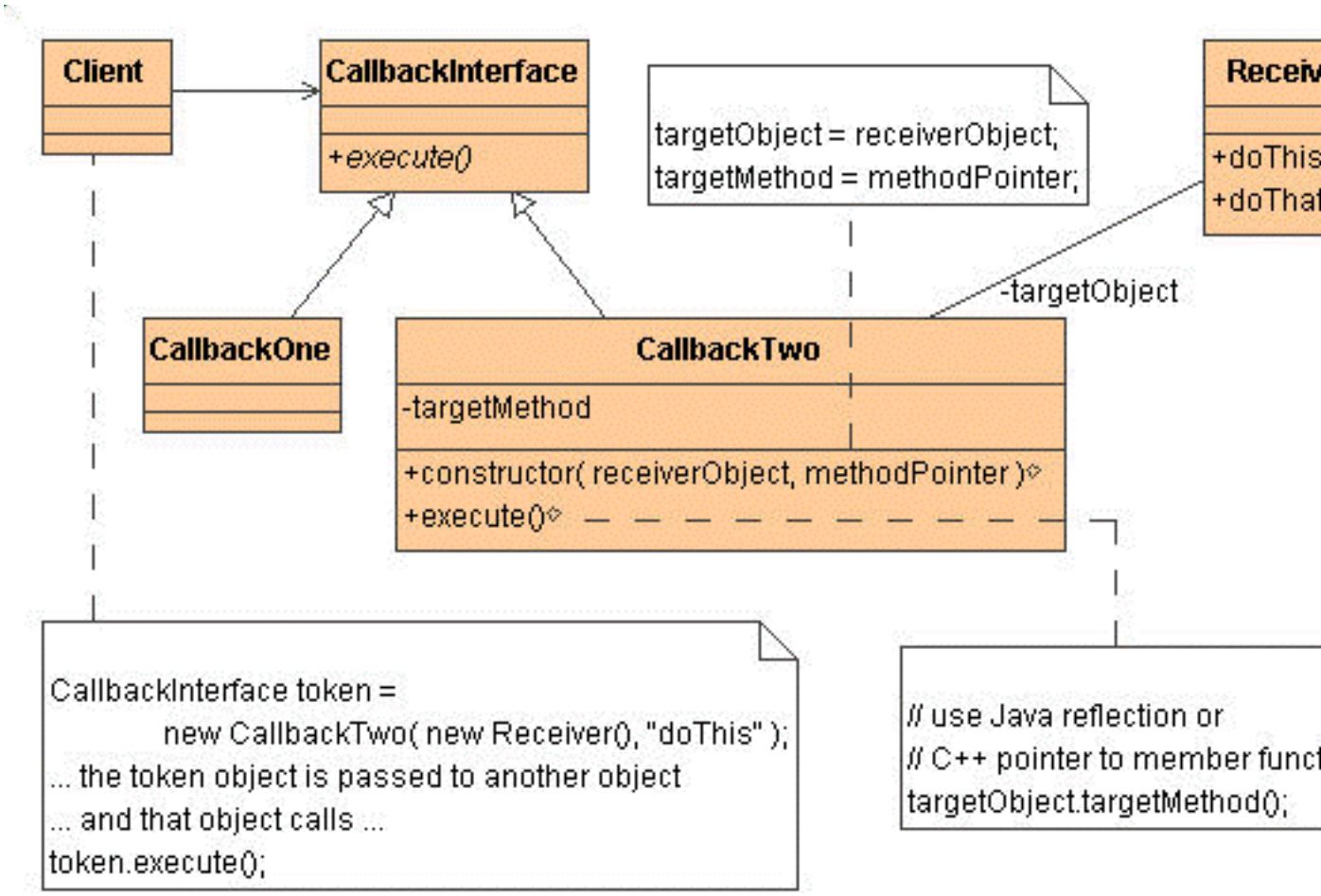


363. Какой паттерн изображён на следующей диаграмме классов?



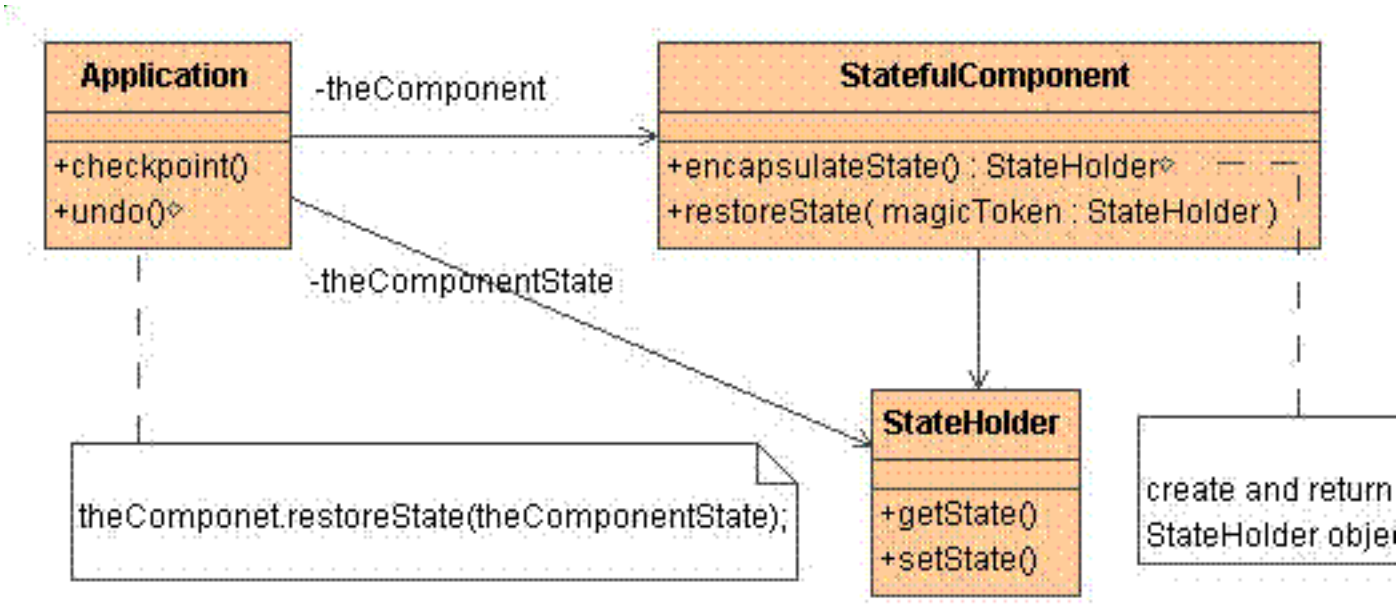
Ответ:
Паттерн Chain of responsibility (Цепочка обязанностей)

364. Какой паттерн изображён на следующей диаграмме классов?



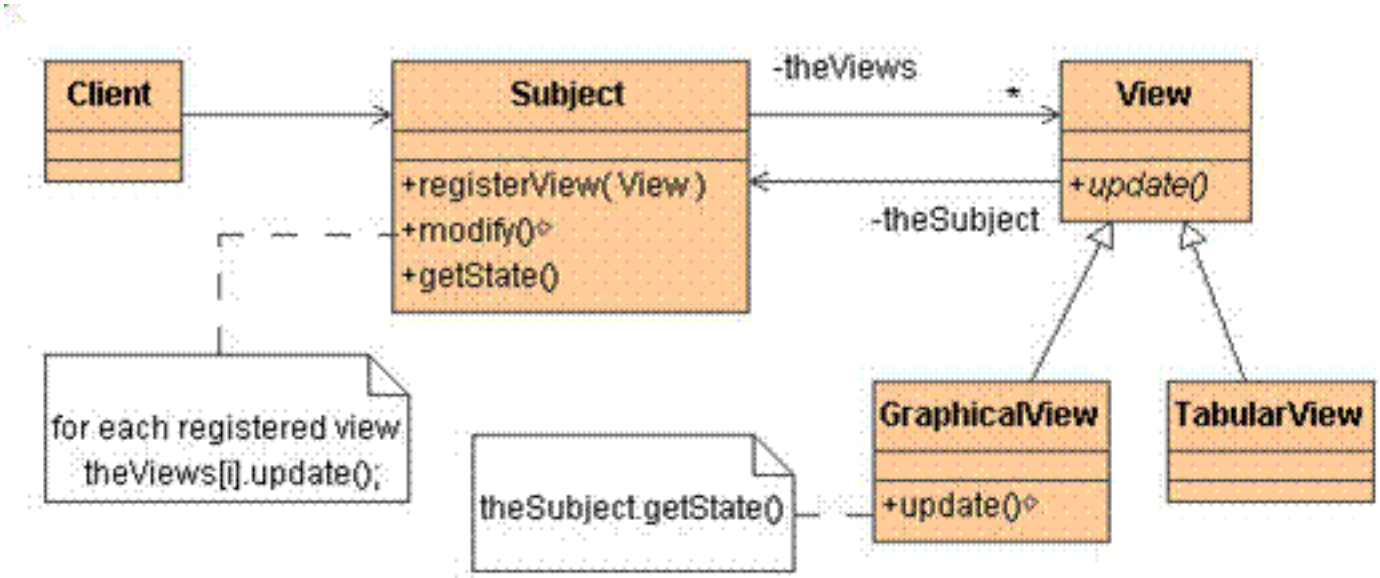
Ответ:
Паттерн Command (Команда)

365. Какой паттерн изображён на следующей диаграмме классов?



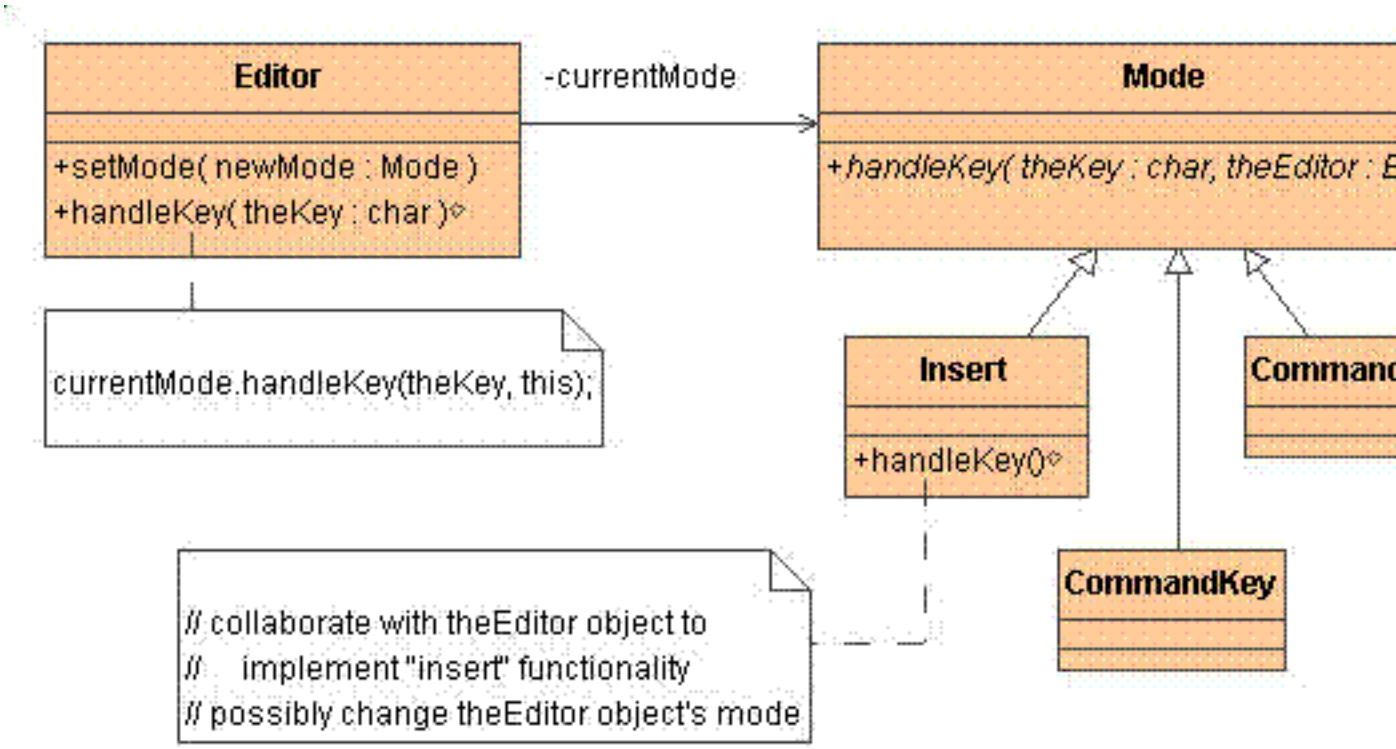
Ответ:
Паттерн Memento (Хранитель)

366. Какой паттерн изображён на следующей диаграмме классов?



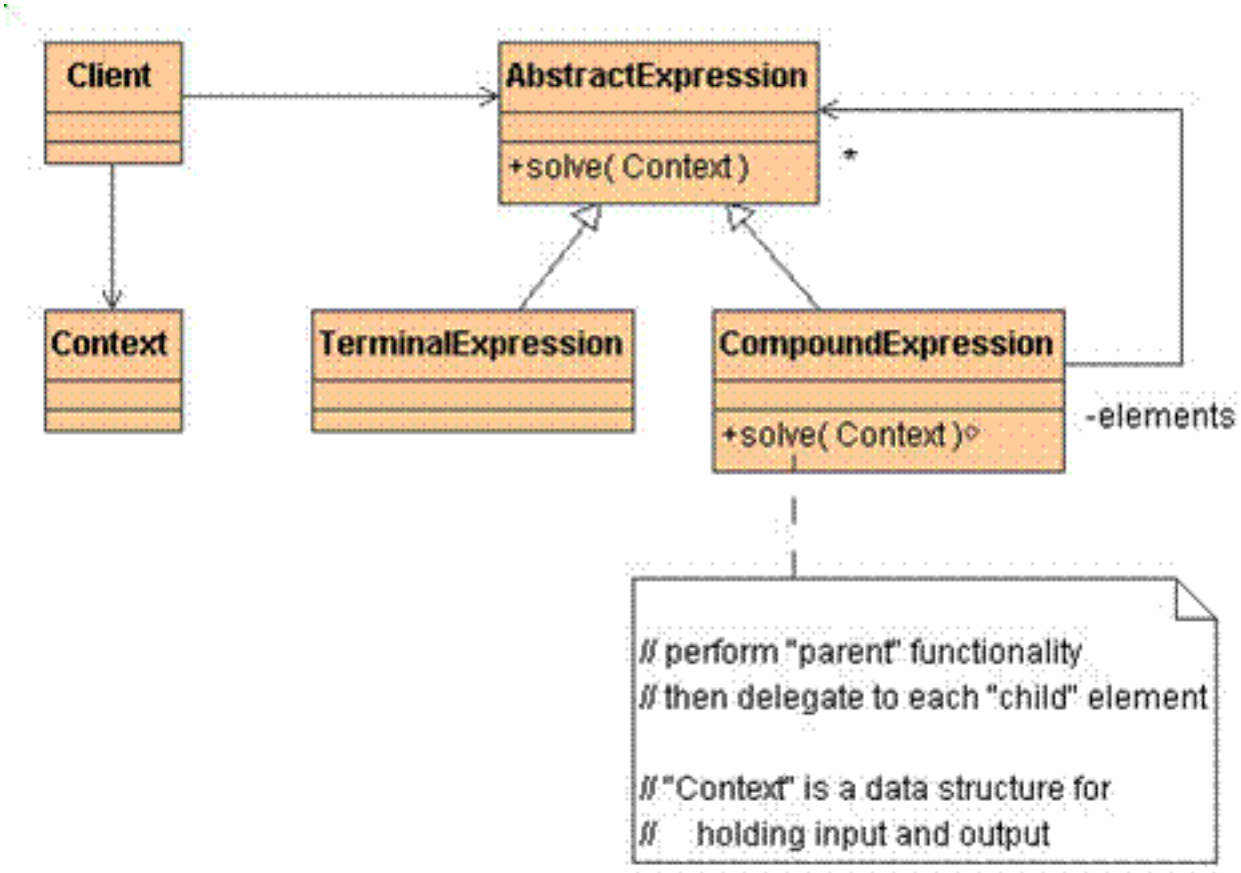
Ответ:
Паттерн Observer (Наблюдатель)

367. Какой паттерн изображён на следующей диаграмме классов?



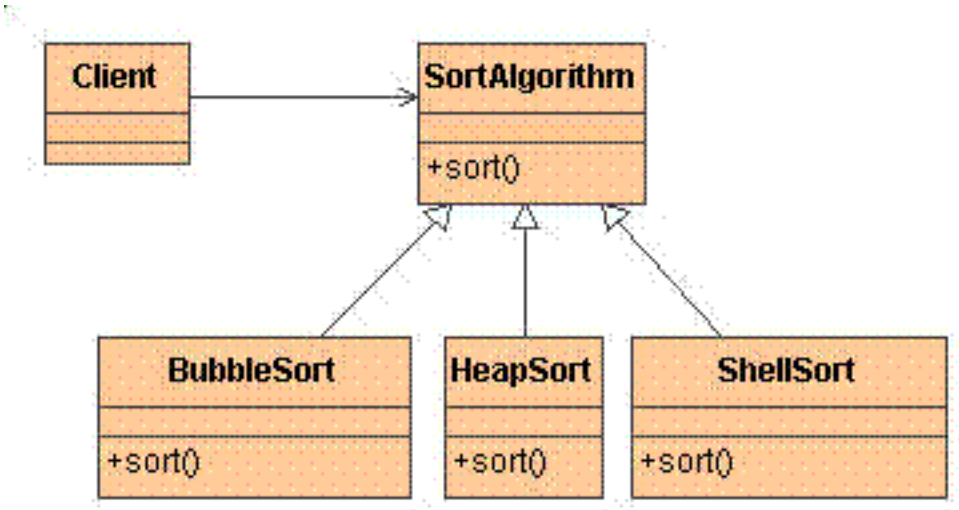
Ответ:
Паттерн State (Состояние)

368. Какой паттерн изображён на следующей диаграмме классов?



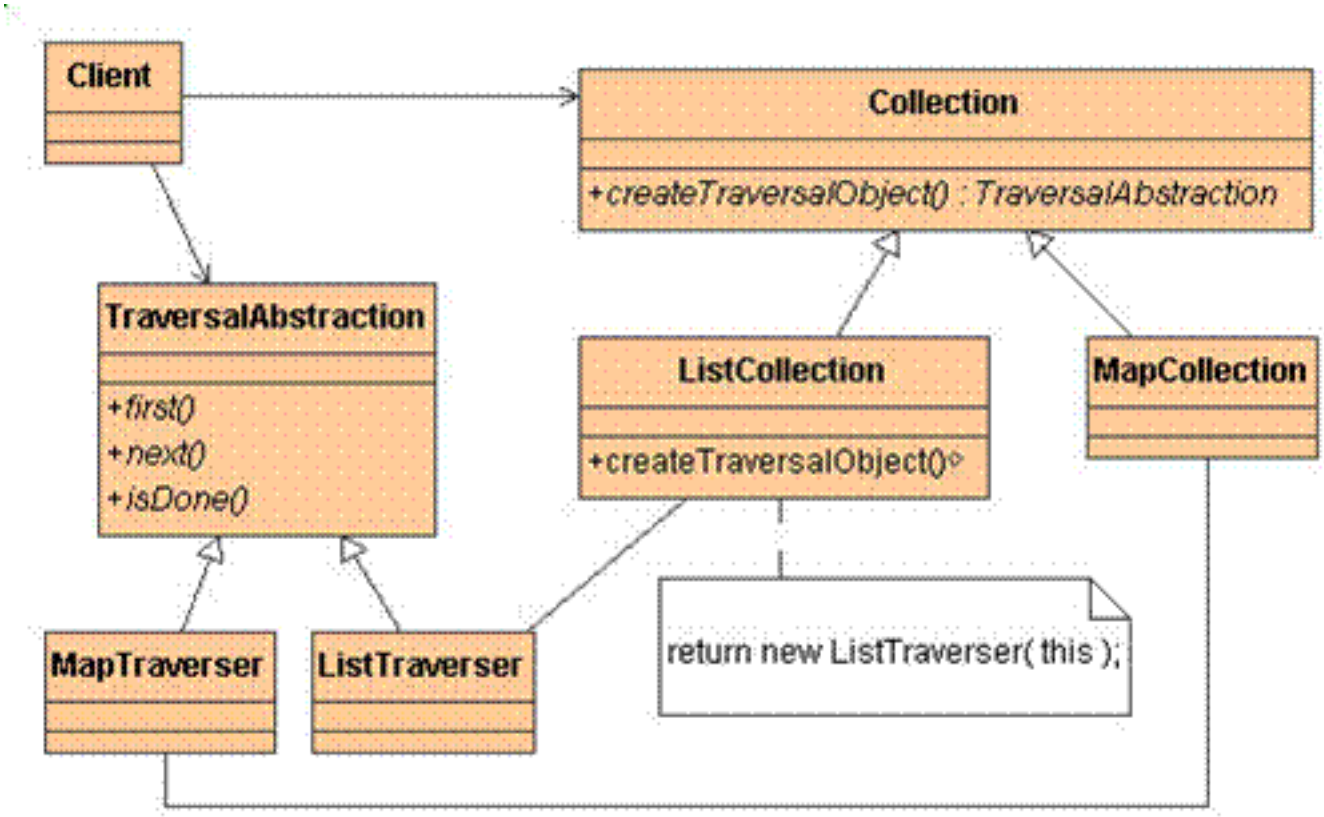
Ответ:
Паттерн Interpreter (Интерпретатор)

369. Какой паттерн изображён на следующей диаграмме классов?



Ответ:
Паттерн Strategy (Стратегия)

370. Какой паттерн изображён на следующей диаграмме классов?

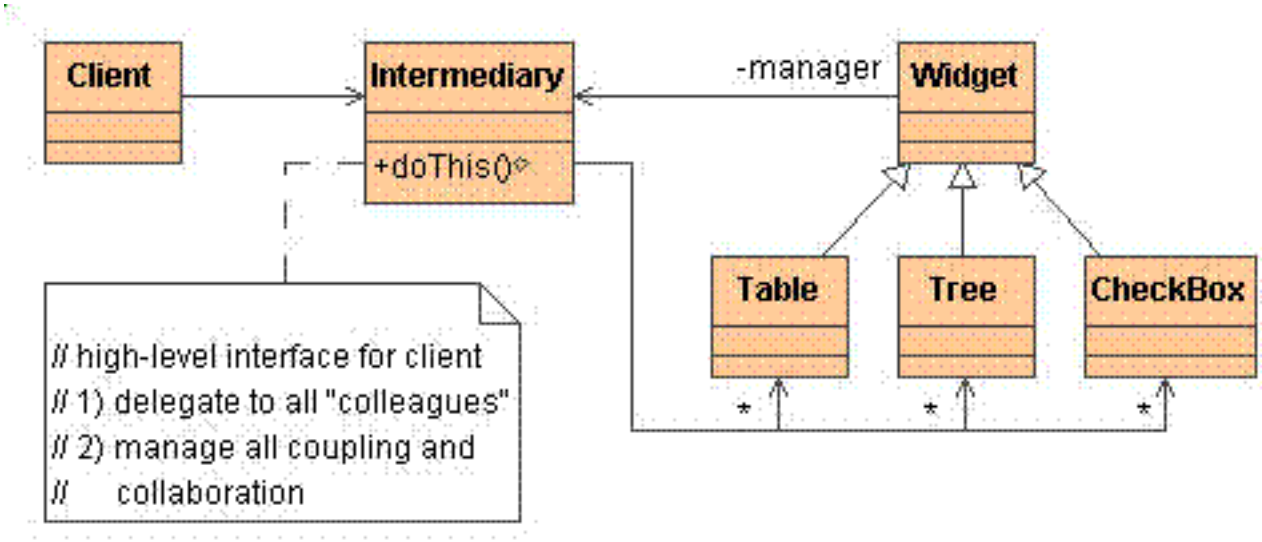


Ответ:
Паттерна Iterator (Итератор)

371. Какой паттерн изображён на следующей диаграмме классов?

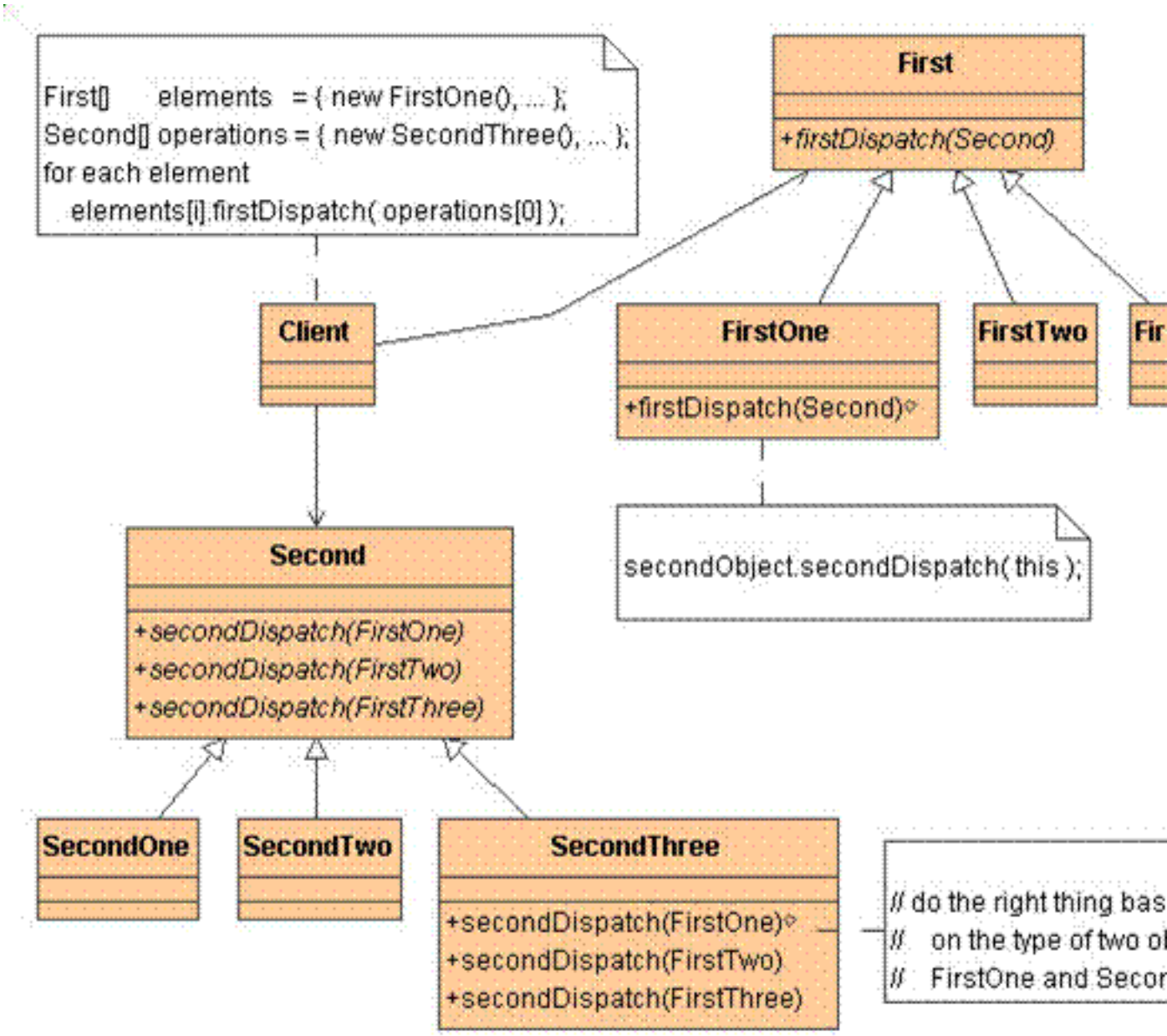
Ответ:
Паттерн Template Method (Шаблонный метод)

372. Какой паттерн изображён на следующей диаграмме классов?



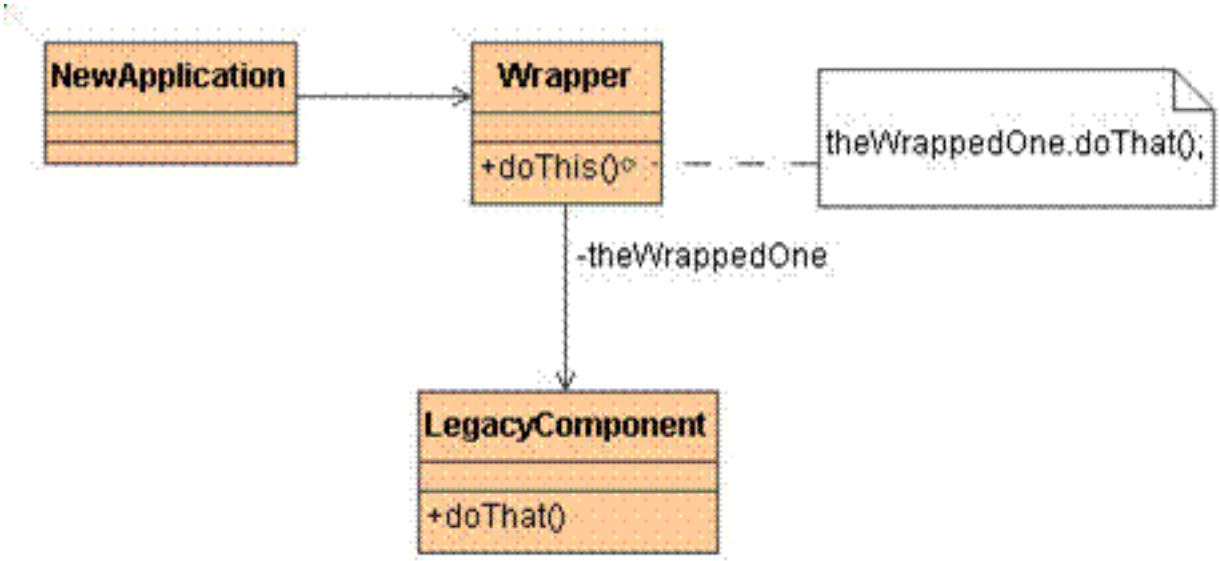
Ответ:
Паттерн Mediator (Посредник)

373. Какой паттерн изображён на следующей диаграмме классов?



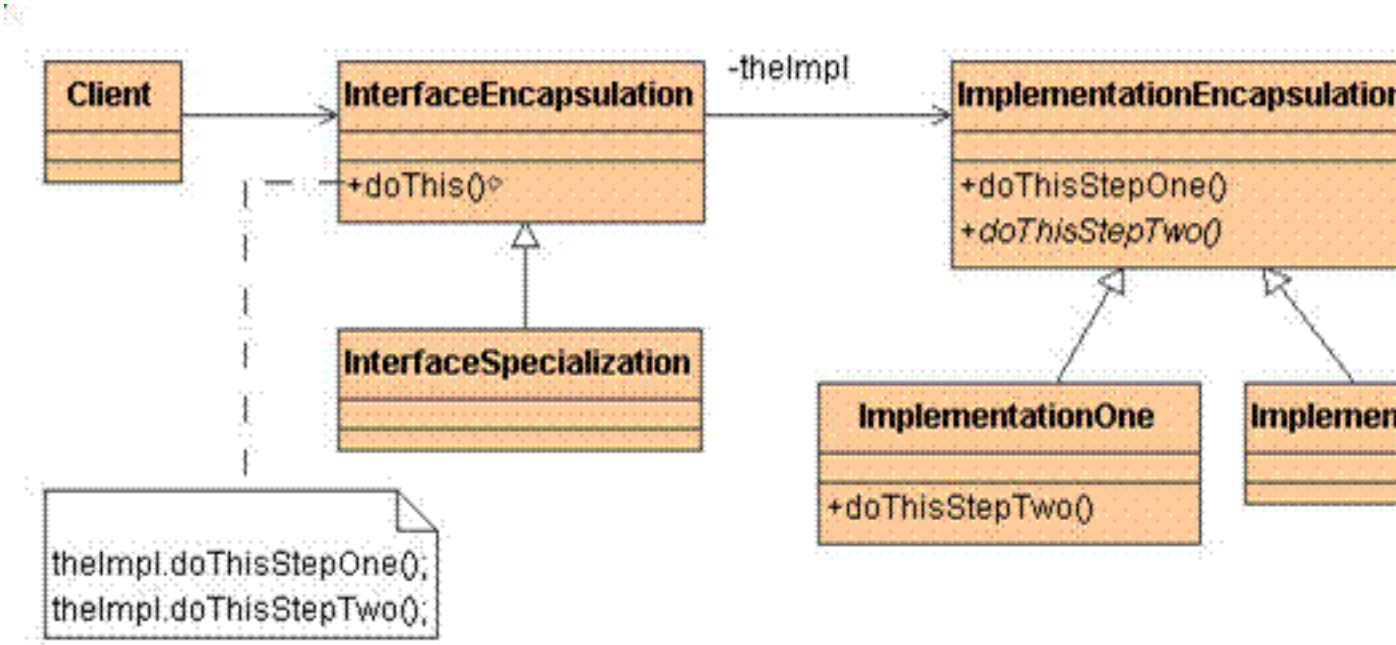
Ответ:
Паттерн Visitor (Посетитель)

374. Какой паттерн изображён на следующей диаграмме классов?



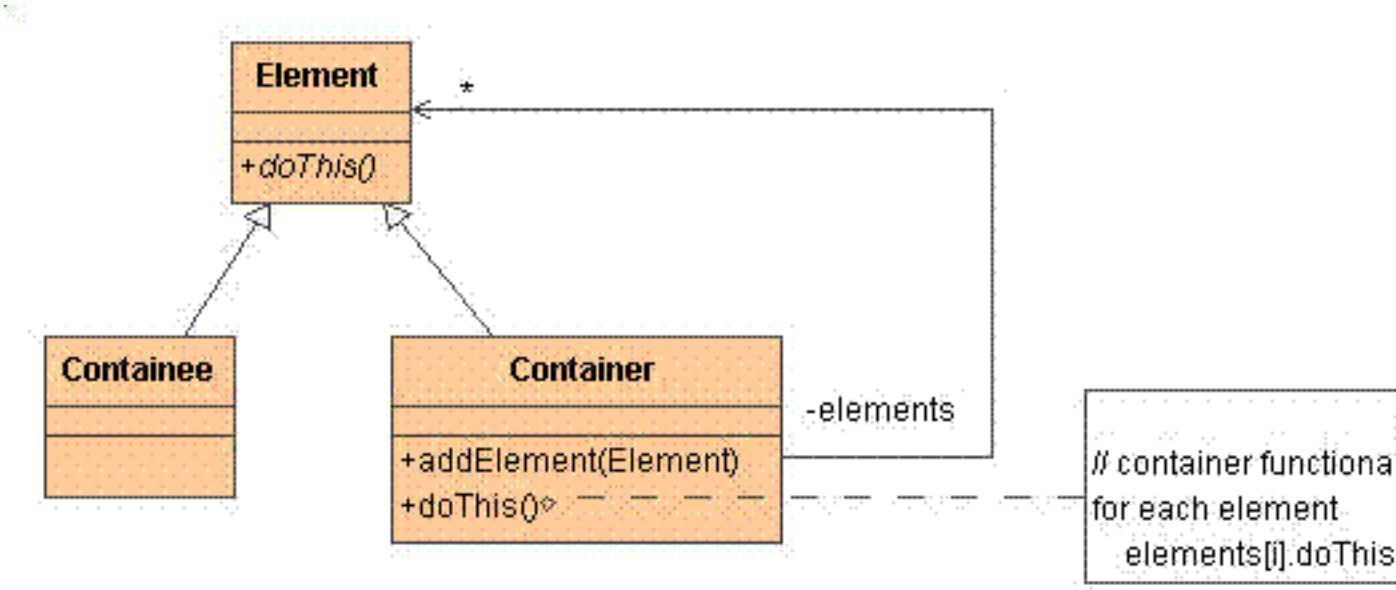
Ответ:
Паттерн Adapter (Адаптер)

375. Какой паттерн изображён на следующей диаграмме классов?



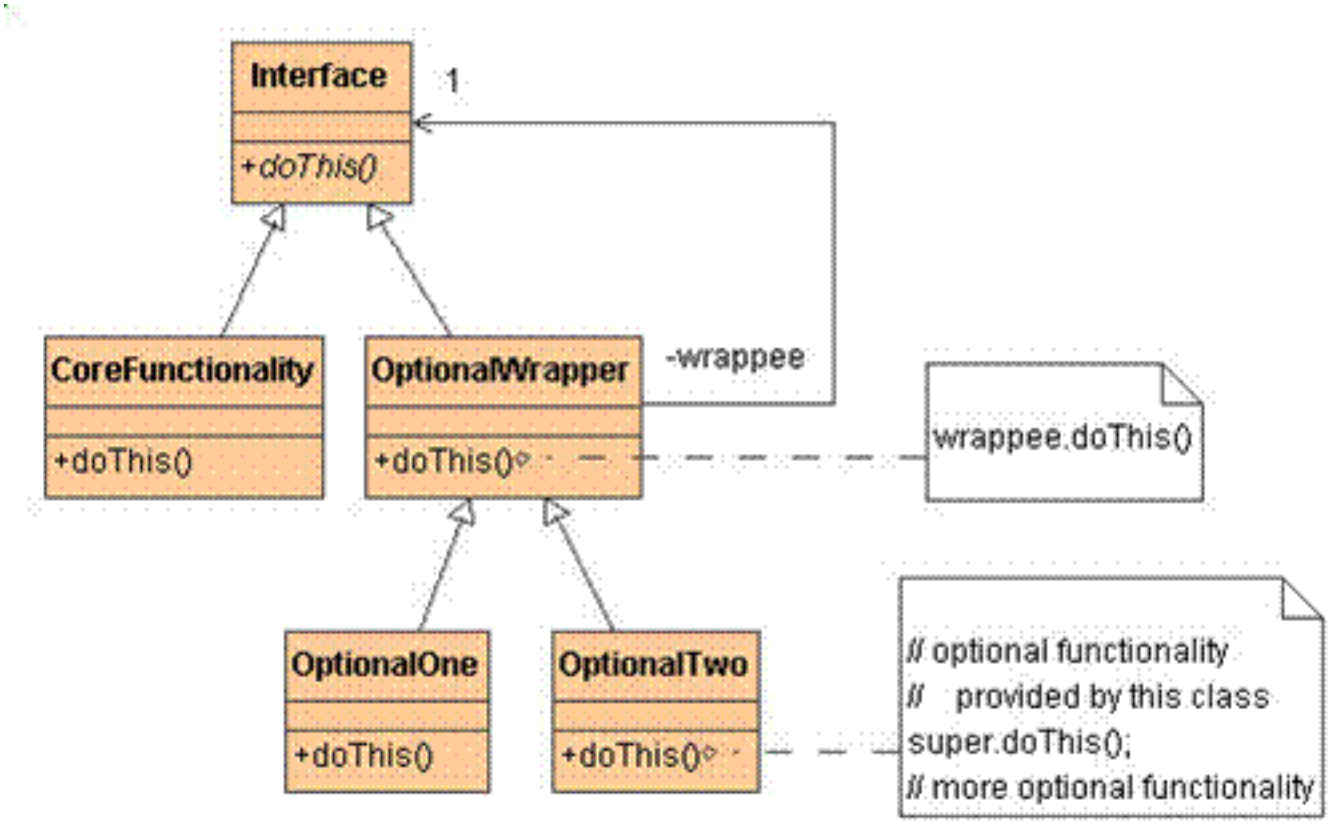
Ответ:
Паттерн Bridge (Мост)

376. Какой паттерн изображён на следующей диаграмме классов?



Ответ:
Паттерн Composite (Компоновщик)

377. Какой паттерн изображён на следующей диаграмме классов?

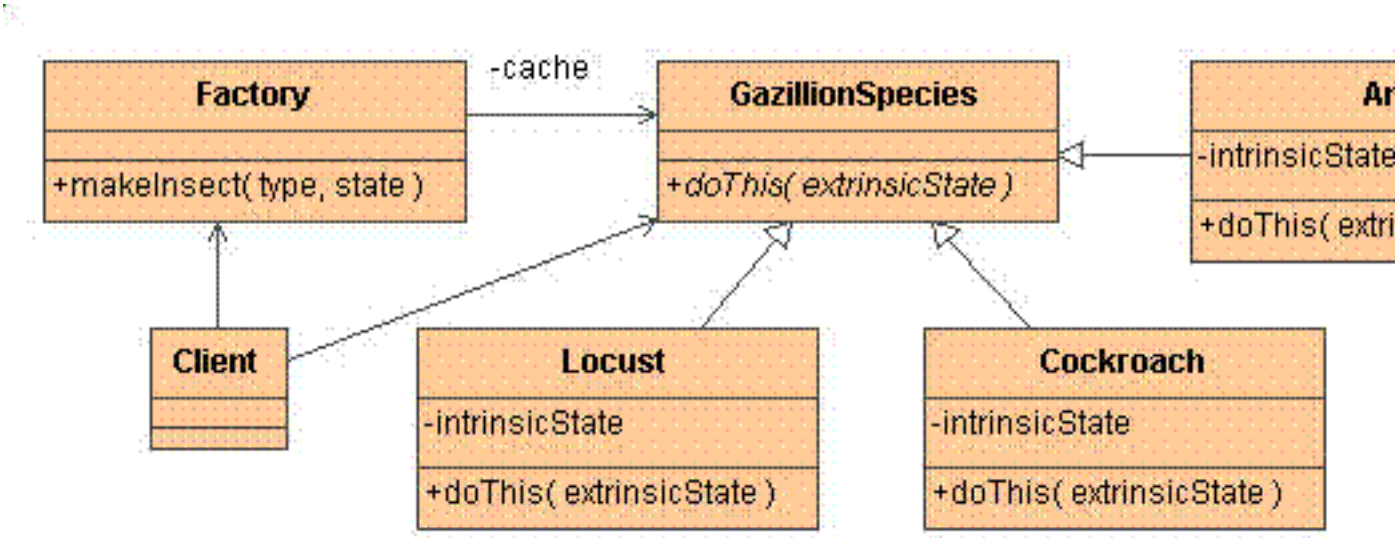


Ответ:
Паттерн Decorator (Декоратор)

378. Какой паттерн изображён на следующей диаграмме классов?

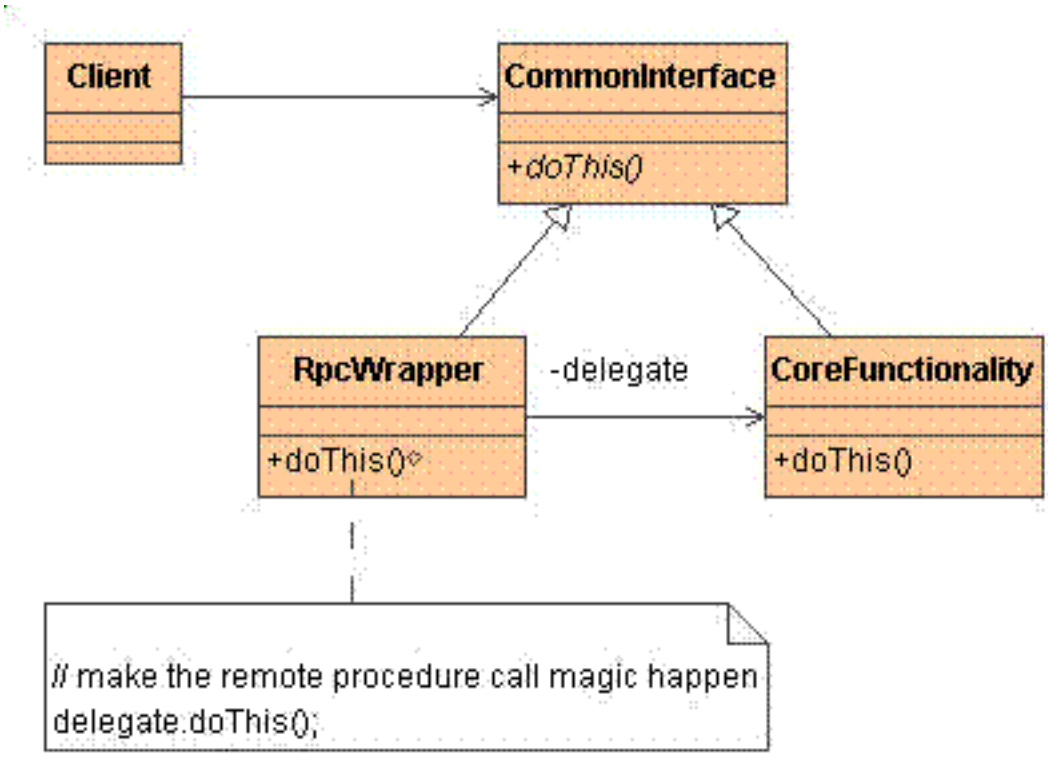
Ответ:
Паттерн Facade (Фасад)

379. Какой паттерн изображён на следующей диаграмме классов?



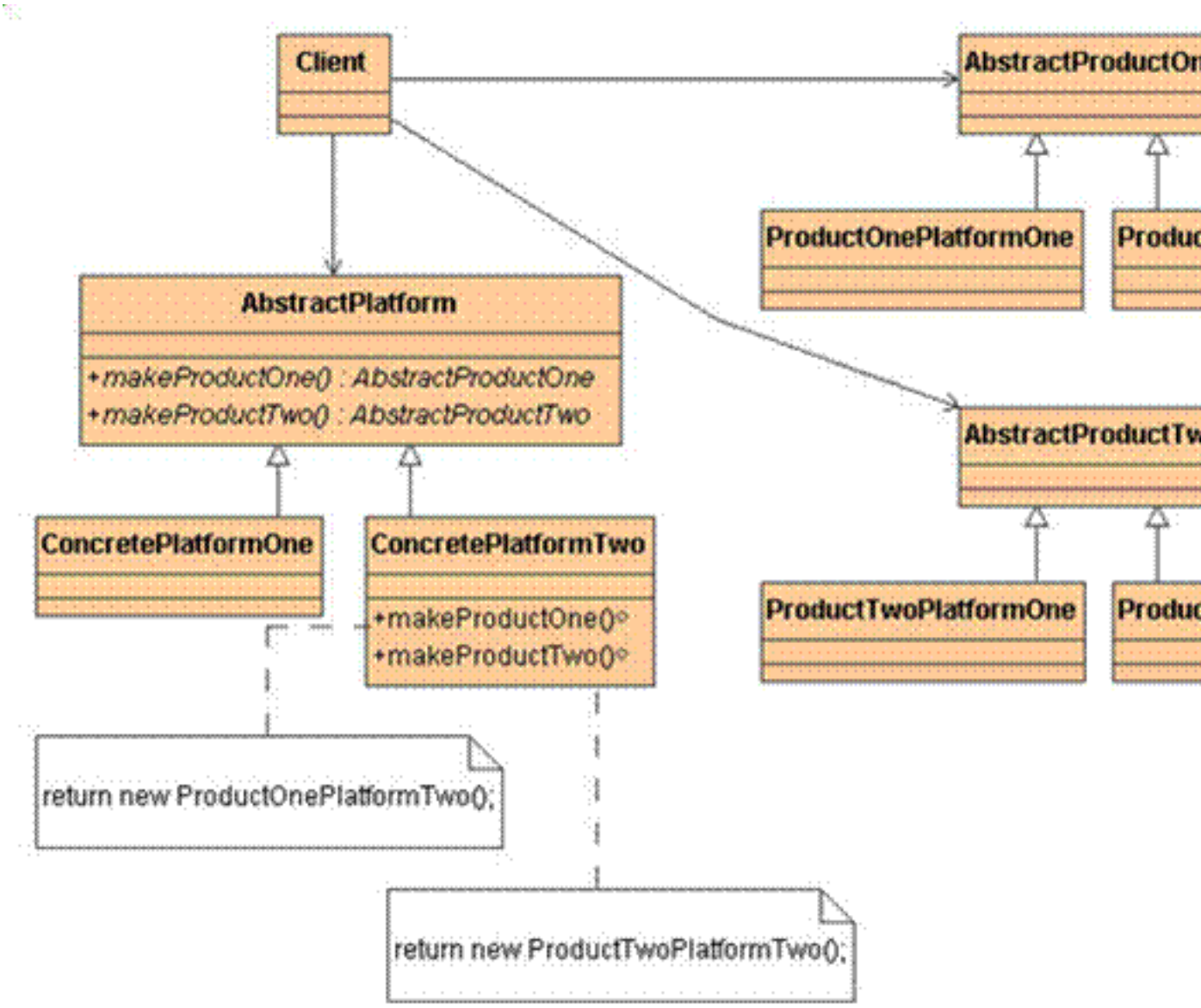
Ответ:
Паттерн Fliweight (Приспособленец)

380. Какой паттерн изображён на следующей диаграмме классов?



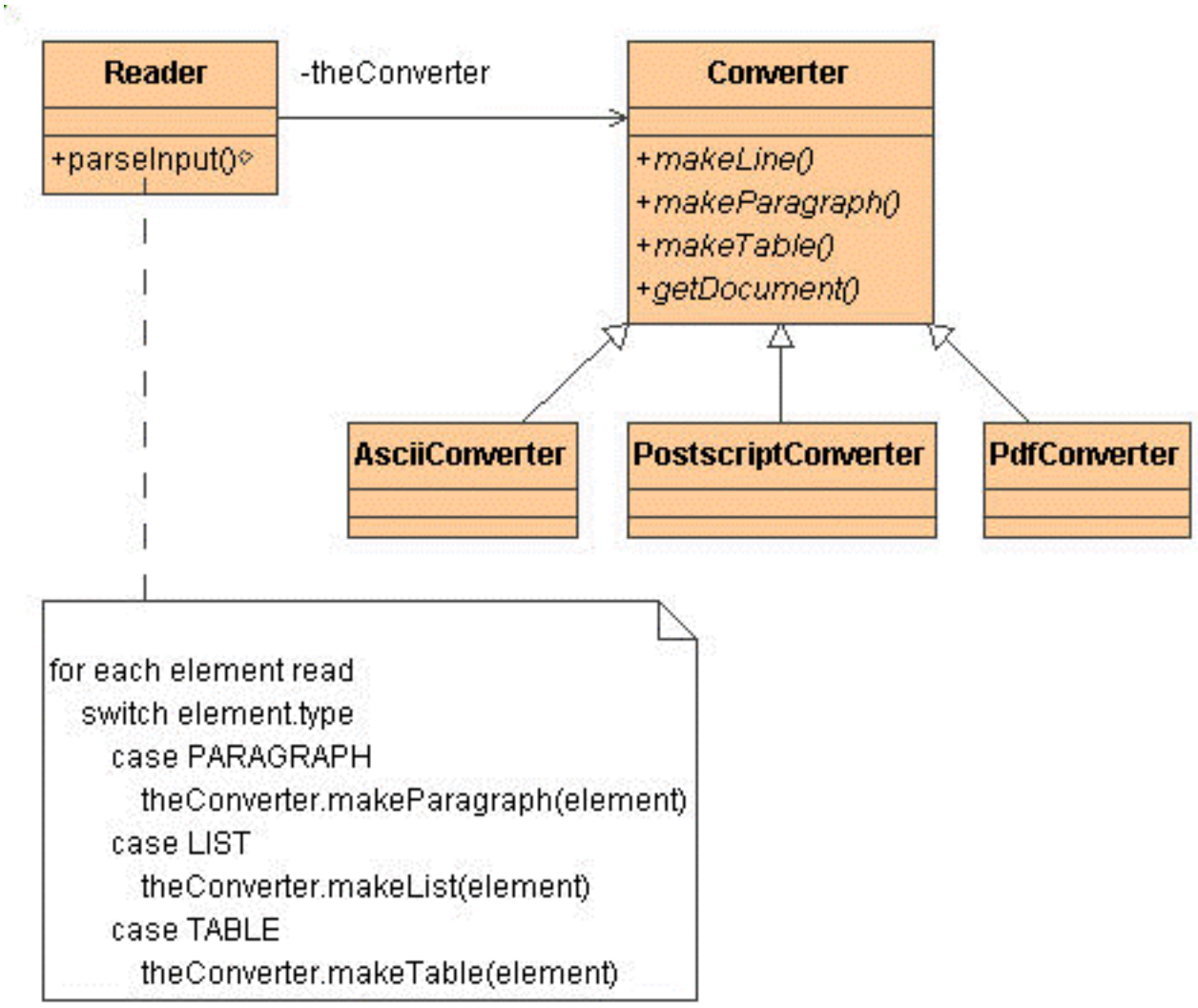
Ответ:
Паттерн Proxy (Прокси)

381. Какой паттерн изображён на следующей диаграмме классов?



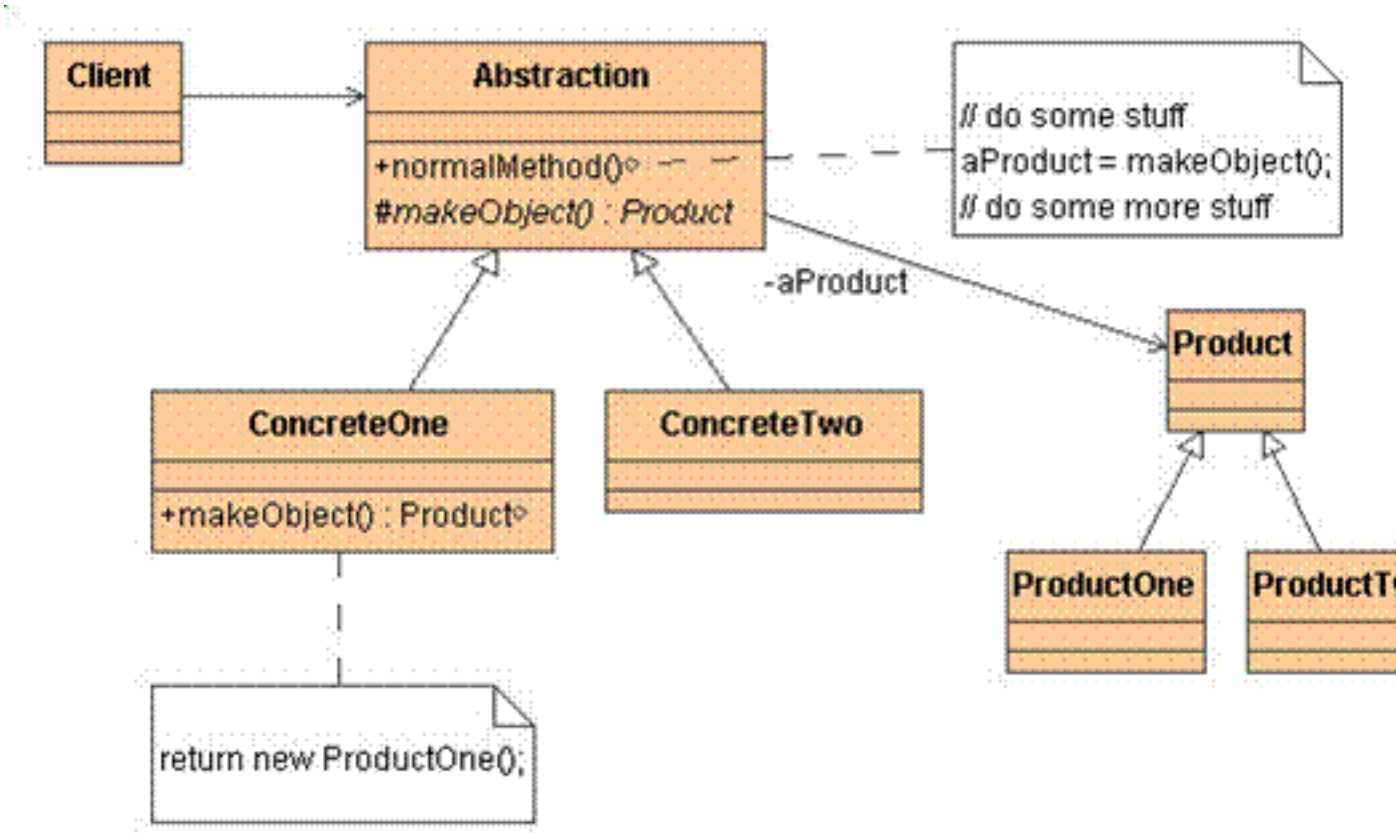
Ответ:
Паттерн Abstract Factory (Абстрактная фабрика)

382. Какой паттерн изображён на следующей диаграмме классов?



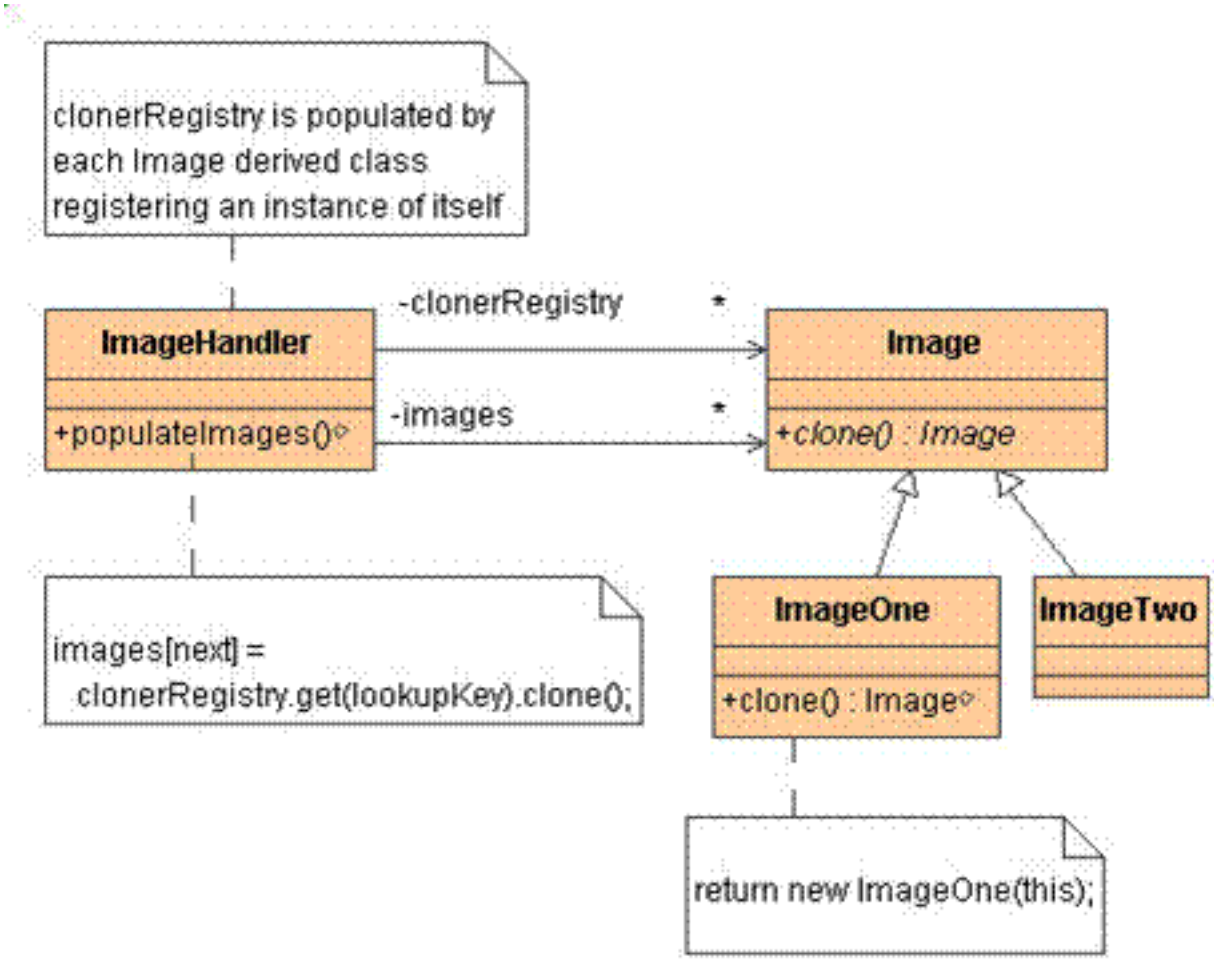
Ответ:
Паттерн Builder (Строитель)

383. Какой паттерн изображён на следующей диаграмме классов?



Ответ:
Паттерн Factory Method (Фабричный метод)

384. Какой паттерн изображён на следующей диаграмме классов?



Ответ:
Паттерн Prototype (Прототип)

385. Какой паттерн изображён на следующей диаграмме классов?

Ответ:
Паттерн Singleton (Одиночка)

386. Что не относится к свойствам Common control class?

Ответ:
Show

387. Какое свойство определяет элементы, видимые для пользователя?

Ответ:
Visible

388. Элемент управления Button может быть активирован:

Ответ:
программно через вызов click event и через нажатие на кнопку

389. Для какого элемента управления доступно свойство CancelButton?

Ответ:
Form

390. Какое из событий не предусмотрено для стандартных элементов управления?

Ответ:
SingleClick

391. Какому элементу управления доступно событие Tick?

Ответ:
Timer

392. Какому элементу управления доступно событие Activated?

Ответ:
Form

393. Какой элемент или элементы управления вызывают всплывающее окно?

Ответ:
Использовать метод Val или CDbl для конвертации текстового значения в числовое.

InputBox и MsgBox

394. Какой из методов элемента управления TextBox не использует буфер обмена?

Ответ:
Clear

395. Какое из свойств элемента управления TextBox рекомендуется изменять в первую очередь?

Ответ:
Name

396. Какие из значений свойства SectionMode элементе управления ListBox допустимы?

Ответ:
None

One

MultiSimple

MultiExtended

397. Установка значения -1 свойству SelectedIndex элементу управления ListBox:

Ответ:
убирает выделение со всех элементов

398. Какой метод элемента управления ListBox удаляет один элемент?

Ответ:
Items.RemoveAt

399. Свойство Items элемента управления ComboBox:

Ответ:
соответствует набору включенных элементов

аналогично свойству Items элемента управления ListBox

имеет методы и свойства

400. Какие значения свойства DropDownStyle элемента управления ComboBox позволяют пользователю вводить значения?

Ответ:
DropDown

401. Объединением каких элементов управления является ComboBox?

Ответ:
ListBox and TextBox

402. Какое количество состояний определено для элемента управления CheckBox?

Ответ:
3

403. Какой стандартный префикс для имени объекта элемента управления CheckBox?

Ответ:
chk

404. Элемент управления CheckBox может соответствовать поведению

Ответ:
Button

405. Какой стандартный префикс для имени объекта элемента управления RadioButton?

Ответ:
rad

406. Какое максимальное количество элементов управления RadioButtons, объединенных в Group Box могут быть одновременно выбрано?

Ответ:
1

407. Какое событие активируется в момент выбора элемента управления RadioButton?

Ответ:
CheckedChanged

408. Как называется элемент управления для показа меню?

Ответ:
MainMenu

409. Какой стандартный префикс для имени элементов MainMenu:

Ответ:
mnu

410. Какой символ позволяет задавать «горячие клавиши» для элементов MainMenu?

Ответ:
&

411. Какой из диалогов не является стандартным?

Ответ:
ZoomDialog

412. В чем причина использования стандартных диалоговых окон?

Ответ:
легкость создания

функциональность знакомая пользователю

413. Какой стандартный префикс для имени объекта диалогового окна:

Ответ:
dlg

414. Какой метод активирует диалоговое окно выбора цвета?

Ответ:
ShowDialog

415. Какой метод активирует диалоговое окно выбора шрифта?

Ответ:
ShowDialog

416. Класс диалогового окна выбора цвета:

Ответ:
ColorDialog

417. Класс диалогового окна выбора шрифта:

Ответ:
FontDialog

418. Какое свойство OpenFileDialog определяет выбранное значение в поле “тип”?

Ответ:
Filter

419. Какие из свойств аналогичны для OpenFileDialog и SaveFileDialog ?

Ответ:
FileName, Filter и InitialDirectory

420. Какое диалоговое окно позволяет увидеть вид документа?

Ответ:
PrintPreview

421. Для какого диалогового окна определено свойство Document?

Ответ:
PrintDialog, PrintPreview и PageSetupDialog

422. Объект какого типа задается в свойстве Document?

Ответ:
PrintDocument

423. Какой элемент управления может вызвать событие PrintPage?

Ответ:
PrintDialog и PrintPreview.

424. Класс поля ввода текста

Ответ:
TextBox

425. Класс списка

Ответ:
ListBox

426. Класс кнопки

Ответ:
Button

427. Свойство Enabled класса Control отвечает за …

Ответ:
доступность элемента управления

428. Свойство Vissible класса Control отвечает за …

Ответ:
видимость элемента управления

429. Метод Hide класса Control управляет свойством …

Ответ:
Visible

430. За полосы прокрутки в элементе управления TextBox отвечает свойство

Ответ:
ScrollBars

431. Элемент управления Label отвечает за

Ответ:
надпись на форме

432. Как настроить элемент управления Button для автоматического изменения размера в соответствии с длиной текста

Ответ:
Необходимо в свойстве AutoSize выбрать значение true

433. Какое свойство элемента управления Button позволяет настроить отображенияе на кнопке значка или растрового изображения

Ответ:
Image

434. На форме не будет стандартных кнопок закрытия, сворачивания и максимизации, если в свойстве формы ControlBox указать значение False

Ответ:
Верно

435. Какой символ позволяет сделать разделительную черту в списке меню

Ответ:
-

436. Как можно добавить горизонтальную полосу прокрутки в элемент управления ListBox

Ответ:
в свойстве MultiColumn указать значение True

437. Значением каакого свойства определяется цвет элемента управления Button

Ответ:
BackColor

438. Вызов какого метода позволит сделать форму невидимой

Ответ:
Hide

439. Объект становится невидимым, если его свойству Visible присвоено значение. . .

Ответ:
False

440. Для какого элемента управления определено свойство Caption

Ответ:
Button

441. Какое значение должно быть задано свойству MultiLine , для того чтобы текстовое поле (элемент управления TextBox) состоял из нескольких строк?

Ответ:
True

442. Чтобы сделать объект незаметным на форме, но поддающимся событию Click, необходимо.

Ответ:
свойству BackStyle пзадать значение Transparent

свойству BorderStyle задать значение None

443. Рамка вокруг метки или графического образа определяется свойством.

Ответ:
BorderStyle

444. Заголовок формы определяется свойством…

Ответ:
Caption

445. При компиляции кода, с опцией _____ код библиотеки включается в код приложения.

Ответ:
статическая линковка библиотеки

446. Опция динамической линковки, предполагает, что код библиотеки _____.

Ответ:
подключается при работе программы, в виде файла

447. При _____ линковке код исполняется быстрее, так как нет необходимости для переадресации вызова, и появляется возможность свободного распространения программы.

Ответ:
статической

448. При статической линковке исполняемый код занимает меньше места на диске и памяти.

Ответ:
Ложь

449. _____ библиотеки — это набор уже скомпилированных подпрограмм или объектов, которые подключаются на стадии линковки к исходной программе в виде объектных файлов.

Ответ:
Статические

450. Выберете строку в которой правильно указано определение экспортируемой переменной из библиотеки dll.

Ответ:
__declspec (dllexport) double var;

451. Выберете строку в которой правильно указано определение экспортируемой функции из библиотеки dll.

Ответ:
__declspec (dllexport) template <class T> bool somefunc(const T&);

452. Выберете строку в которой правильно указано определение импортируемой функции из библиотеки dll.

Ответ:
__declspec (dllimport) unsigned int GETinfo(char*);

453. Выберете строку в которой правильно указано определение импортируемой переменной из библиотеки dll.

Ответ:
__declspec (dllimport) char* USER;

454. В приведенном ниже коде приведен код заголовочного файла динамической библиотеки и часть кода из исходного файла “disk.cpp”.

Для каких целей применят код файла “disk.cpp”?

```
/// Содержимое файла diskfree.h
#ifdef _DISKFREE_H_
#define _DISKFREE_H_

#ifndef __DISKFREE__
#define DISKFREELIB    __declspec(dllexport)
#else
#define DISKFREELIB    __declspec(dllimport)
#endif

DISKFREELIB unsigned int DiskFree(unsigned int);

#endif

/// В файле исходного текста disk.cpp
...
#define __DISKFREE__
#include "diskfree.h"
...
```

Ответ:
Файл “disk.cpp” нужен для использования функции dll библиотекти

455. В приведенном ниже коде приведен код заголовочного файла динамической библиотеки и часть кода из исходного файла “disk.cpp”.

Для каких целей применят код файла “disk.cpp”?

```
/// Содержимое файла diskfree.h
#ifdef _DISKFREE_H_
#define _DISKFREE_H_

#ifndef __DISKFREE__
#define DISKFREELIB    __declspec(dllexport)
#else
#define DISKFREELIB    __declspec(dllimport)
#endif

DISKFREELIB unsigned int DiskFree(unsigned int);

#endif

/// В файле исходного текста disk.cpp
...
#include "diskfree.h"
...
```

Ответ:
Файл “disk.cpp” нужен для создания dll библиотекти

456. При установке DLL принято помещать в папку Windows либо Windows\System, также допустимо использовать папку проекта. В старых версиях до Windows 2000 нужно было следить за тем, что бы в этих папках не оказались разные версии одного и того же модуля dll.

Ответ:
Истина

457. Динамическую линковку во время загрузки (load-time dynamic linking) также называют _____ .

Ответ:
Неявная (статическая) линковка

458. Операционная система выполняет загрузку dll библиотеки при загрузке кода приложения, можно утверждать, что это _____.

Ответ:
Неявная (статическая) линковка

459. Исполняемый код выполняет вызов ф. DLL явно загружая DLL, используя функции загрузки LoadLibrary().

Ответ:
Явная (динамическая) линковка

460. Исполняемый код осуществляет доступ к библиотечным функциям через указатель на функцию.

Ответ:
Явная (динамическая) линковка

461. Для одной и той же библиотеки возможны оба метода доступа, одно приложение может явно линковать DLL, другое неявно.

Ответ:
Истина

462. При явной линковке, исполняемый код использующий DLL должен включать заголовочный файл с экспортируемыми функциями и C++ классами, для каждого файла в котором выполняется вызов ф. из DLL.

Ответ:
Ложь

463. DLL позволяет сделать общим ресурс. Множество приложений может одновременно использовать содержимое единственной копии DLLв памяти.

Ответ:
Истина

464. Недостаток применения динамической линковки состоит в том, что наличие DLL определит возможность запуска программы.

Ответ:
Истина

465. Преимущество применения динамической линковки обеспечение поддержки после продажи. Например, драйвер дисплея может быть изменен, что бы поддерживать новые модели, которых не было на момент продажи приложения.

Ответ:
Истина

466. Преимущество применения динамической линковки упрощение создания международной версии ПО, размещая ресурсы в DLL легко создать такое приложение.

Ответ:
Истина

467. Планирование отдельных деталей системы и пишется код осуществляется на стадии …

Ответ:
построение

468. Что может происходить время от времени в Унифицированном процессе:

Ответ:
все верно

469. Определения методов:

Ответ:
не следует помещать в заголовочные файлы

вероятно, не должны предоставляться пользователям

470. В диаграмме последовательностей:

Ответ:
горизонтальные стрелки представляют собой сообщения

вертикальные пунктирные линии представляют собой линию жизни

471. В диаграмме последовательностей:

Ответ:
горизонтальные стрелки представляют собой сообщения

вертикальные пунктирные линии представляют собой линию жизни

472. Защита данных от несанкционированного доступа другими функциями называется

Ответ:
сокрытием данных

473. Библиотечная функция getche():

Ответ:
печатает на экране символ, соответствующий нажатой клавише

возвращает символ в случае нажатия какой-либо из клавиш

474. Оператор goto вызывает переход на:

Ответ:
метку

475. Наиболее важным из назначений функции является:

Ответ:
обработка аргументов и возвращение значения

476. Какие из перечисленных ниже элементов программы можно передавать в функцию:

Ответ:
константы

переменные

структуры

477. Перегруженные функции:

Ответ:
являются группой функций, имеющих одно и то же имя

облегчают процесс программирования

478. Значение аргумента по умолчанию:

Ответ:
может использоваться вызывающейся программой

может использоваться функцией

479. Методу класса всегда доступны данные:

Ответ:
объекта, членом которого он является

480. Константный метод, вызванный для объекта класса,

Ответ:
не может изменять как неконстантные, так и константные поля

481. Поток C++:

Ответ:
представляет собой поток данных из одного места в другое

ассоциирован с конкретным классом

482. В файл проекта включаются:

Ответ:
данные о содержимом файлов, входящих в проект

даты последних изменений файлов, входящих в проект

483. Глобальная переменная определена в файле A. Чтобы получить доступ к ней из файла B, необходимо:

Ответ:
объявить ее в файле B, используя extern

484. Шаблоны позволяют удобным способом создавать семейства:

Ответ:
функции

классов

485. Использование typedef позволяет:

Ответ:
менять имена типов

менять названия классов

486. Пусть имеется ассоциативная связь между классами A и B. Пусть objA - объект класса A, а objB - объект класса B. Тогда:

Ответ:
objA может послать сообщение objB

objB может помочь objA выполнить задачу

487. Операция, выполняющая заданные действия над пользовательским типом данных, называется:

Ответ:
перегруженной

488. Переменная, описанная внутри блока, видима:

Ответ:
от точки своего объявления до конца блока

489. Общая архитектура системы планируется на стадии

Ответ:
развитие

490. Разделение программы на функции:

Ответ:
упрощает представление программы

сокращает размер программного кода

491. При создании нового класса на диаграмме последовательностей:

Ответ:
рисуется прямоугольник с его именем на соответствующей высоте

начинается его линия жизни

492. Операция отношения:

Ответ:
имеет своим результатом булево значение

сравнивает значения двух операндов

493. Аргументы командной строки

Ответ:
набираются после названия программы в командной строке

делаются доступными с помощью аргументов main()

494. Общедоступная часть библиотеки классов обычно содержит

Ответ:
объявления методов

объявления классов

495. Пространства имен используются для:

Ответ:
сужения области видимости элементов программы

496. Водопадный процесс:

Ответ:
состоит из различных этапов

может протекать только в одном направлении

497. Библиотечная функция exit() предназначена для выхода из:

Ответ:
программы, в которой она содержится

498. Чтобы определять объекты класса в разных файлах, в каждом из них необходимо:

Ответ:
определять класс

499. Возможности будущей программы и ее осуществимость выявляются на стадии …

Ответ:
начало

500. Варианты использования (кроме всего прочего) нужны для:

Ответ:
обеспечения выбора подходящих атрибутов класса

определения того, какие классы необходимы в программе

501. Вариант использования - это, на самом деле -

Ответ:
действие

502. Описание вариантов использования иногда пишется в двух

Ответ:
колонках

503. Действующим субъектом может быть

Ответ:
некая система, взаимодействующая с нашей

человек, взаимодействующий с разрабатываемой системой

504. Классы в программе могут соответствовать:

Ответ:
существительным в описаниях вариантов использования

505. В качестве образца по отношению к объекту выступает:

Ответ:
класс

506. Какие из перечисленных ниже причин являются главными для использования объектно-ориентированных языков?

Ответ:
возможность создания собственных типов данных

объектно-ориентированные программы легче концептуализируются

507. Если язык обеспечивает возможность создания пользовательских типов данных, то говорят, что язык называется:

Ответ:
расширяемым

508. Универсальный язык моделирования - это:

Ответ:
средство визуализации организации программы

вспомогательное средство при разработке программного обеспечения

509. Оператор break производит выход:

Ответ:
из цикла или ветвления наибольшей глубины вложенности

510. Структура объединяет:

Ответ:
логически связанные данные

переменные

511. Перечисление объединяет:

Ответ:
именованные целые числа

512. Аргумент функции - это:

Ответ:
значение, передаваемое вызывающей программой в функцию

513. Статическая локальная переменная используется для:

Ответ:
ограничения области видимости переменной до одной функции

сохранения значения переменной после завершения функции

514. Разбивать программу на несколько файлов желательно, потому что:

Ответ:
все верно

515. Так называемые скрытые файлы библиотеки классов:

Ответ:
могут быть сделаны доступными с помощью дружественных функций

могут состоять только из объектного кода

516. Унифицированный процесс разработки ПО разбивается на этапы:

Ответ:
передача

начало

развитие

построение