ECE 462 Midterm Exam 1

10:30-11:20AM, September 21, 2007

1 Template Classes and the STL Library

1.1 Container Classes

Which statement is correct?

Answer: B

A. An element can be inserted anywhere in a stack.

B. An element can be inserted anywhere in a list.

C. In a map, the values must be unique and the keys can be the same. Thus, it can be a one-to-many mapping.

D. In a map, the values and the keys must be unique. Thus, it must be a one-to-one mapping.

E. Once a vector object is created, the first element cannot be deleted.

1.2 Container Classes

Which statement is correct?

Answer: C

A. Elements can be inserted or removed only at the end of a list, not anywhere else.

B. Elements can be inserted only at the end of a list and removed at the beginning of a list.

C. In a Java list, duplicate elements are allowed.

D. In a C++ list, duplicate elements are not allowed. The elements must be unique.

E. Elements can be inserted or removed only at the beginning of a list.

1.3 Template in C++

Which statement is correct?

Answer: E

A. A template can be used in built-in container classes only. A programmer cannot create a class with a template.

B. A template can be replaced by a primitive type (int, char, double ...) only. C++ does not allow replacing a template by a class.

C. A template can be replaced by a class only. C++ does not allow replacing a template by a primitive type.

D. A class can have only one template and the placeholder must be called `T`.

E. A class can have several templates and they can be replaced by different types.

1.4 Java Vector and Class Hierarchy

What is the output of this program?

Answer: B

```
import java.util.*;
class Base {
    protected int b_val;
    public Base(int val) { b_val = val; }
    public void print() { System.out.println("B " + b_val); }
}
class Derived extends Base {
    public Derived(int val) { super(val); }
    public void print() { System.out.println("D " + b_val); }
}
class Program {
    public static void main( String[] args )
    {
        Vector<Base> objectvec = new Vector<Base>();
        Base [] objects = new Base[4];
        objects[0] = new Base(0);
        objects[1] = new Derived(1);
        objects[2] = new Base(2);
        objects[3] = new Derived(3);
        objectvec.add(objects[0]);
        objectvec.add(objects[1]);
        objectvec.add(objects[2]);
        objectvec.add(objects[3]);
        Iterator iter = objectvec.iterator();
        Base iterobj;
        while ( iter.hasNext() ) {
            iterobj = (Base) iter.next();
            iterobj.print();
        }
    }
}
```

A. B0 B1 B2 B3

B. B0 D1 B2 D3

C. B0 D0 B0 D0

D. B1 D0 B3 D2

E. D0 D1 D2 D3

1.5 Container and Copy Constructor

How many times is

```
X (const X & xin) { p = xin.p; cout << "X(const X&)" << endl; }
```

called in this program?

Answer: E (12)

```
#include <iostream>
#include <vector>
using namespace std;
class X {
int p;
public:
   X( int q ) { p = q; cout << "X(int)" << endl; }
   X (const X & xin) { p = xin.p; cout << "X(const X&)" << endl; }
   int getp() const { return p; }
   void changeState( int pp ) { p = pp; }
};
void print( vector<X> );
int main()
{
   vector<X> vec;
   X x1( 2 );
   X x2( 3 );
   X x3( 5 );
   vec.push_back( x1 );
   vec.push_back( x2 );
   vec.push_back( x3 );
   print( vec );
   x2.changeState( 1000 );
   print( vec);
   return 0;
}
void print( vector<X> v ) {
   cout << "\nvector size is: " << v.size() << endl;
   vector<X>::iterator p = v.begin();
   while ( p != v.end() )
      cout << (*p++).getp() << " ";
   cout << endl << endl;
}
```

A. 0 B. 3 C. 6

D. 7 E. none of the above

1.6 C++ Set

What is the output of this program?

Answer: `4 c d f g` *(The letters must be ordered.)*

```
#include <iostream>
#include <set>
#include <string>
using namespace std;
int main(void)
{
  set<string> charset;
  set<string>::iterator iter;
  charset.insert("c");
  charset.insert("g");
  charset.insert("d");
  charset.insert("f");
  charset.insert("c");

  cout << charset.size() << endl;
  for (iter = charset.begin();iter != charset.end(); iter++)
    { cout << *iter << " "; }
  cout << endl;

  return 0;
}
```

2 Inheritance and Polymorphism

2.1 Inheritance and Class

Which statement is correct?

Answer: B

A. A base class must not be an abstract class.

B. A derived class has everything of the base class and probably additional attributes and methods.

C. Polymorphism means objects can protect their private data.

D. The only purpose of creating a class is to encapsulate data.

E. In Java, a private attribute can be seen by derived classes; a protected attribute cannot be seen by derived classes.

2.2 Inheritance, Interface, and Implementation

Answer: D

Which statement is correct?

A. An abstract class cannot have any attribute.

B. If a C++ class has a virtual function, this class is abstract.

C. If a class has no virtual function, this class cannot have derived classes.

D. A class provides an interface through which messages can be sent.

E. If an object X is not declared as an attribute inside a class Y, this object X is not allowed to send any message to any object of class Y.

2.3 Programming Assignment 1

When you create a derived class from `JPanel` and you create a new implementation of the `paintComponent` method, which is the best description of the concept?

Answer: A

A. Polymorphism. Since this derived class overrides the implementation, this new implementation is called.

B. Encapsulation. This `paintComponent` method is not directly called anywhere by your code. Encapsulation means the caller of a function is invisible to a programmer.

C. Inheritance. This `paintComponent` method is inherited from the JPanel class.

D. Abstraction. JPanel is an abstract class and no object can be created for JPanel.

E. Containment. All painting is accomplished by this derived class.

2.4 Inheritance and Polymorphism

What is the output of this program?

Answer: D

```
class X
{
    public void print() { System.out.print("X"); }
}
class Y extends X
{
    public void print() { System.out.print("Y"); }
}
class Z extends Y
{
}
class prog
{
    public static void main (String[] args)
    {
        X [] objs;
        objs = new X[3];
        objs[0] = new X();
        objs[1] = new Y();
        objs[2] = new Z();
        for (int ocnt = 0; ocnt < 3; ocnt ++)
            {
                objs[ocnt].print();
            }
    }
}
```

A. This program does not compile.

B. XXX

C. YYY

D. XYY

E. XYZ

2.5 Inheritance and Polymorphism

What is the output of this program? If the program cannot compile, write "cannot compile".

Answer: 1 2 4

```
#include <iostream>
using namespace std;
class BaseC {
public:
  int val;
  BaseC() { val = 2; }
  virtual void addObj(BaseC * obj) {
    cout << (obj -> val) << " ";
  }
};
class DerivedC: public BaseC {
public:
  DerivedC() { val = 1; }
  virtual void addObj(BaseC * obj) {
    cout << 2 * (obj -> val) <<  " ";
  }
  virtual void addObj(DerivedC * obj) {
    cout << 3 * (obj -> val) <<  " ";
  }
};

int main(int argc, char * argv[]) {
  BaseC * obj1 = new BaseC;
  BaseC * obj2 = new DerivedC;
  DerivedC * obj3 = new DerivedC;
  obj1 -> addObj (obj2);
  obj2 -> addObj (obj3);
  obj3 -> addObj (obj1);
}
```

2.6 Inheritance and Polymorphism

Which statement is correct?

Answer: A

A. In C++, polymorphism is achieved using virtual functions.

B. Polymorphism allows objects to send messages to each other.

C. An object of the derived class "has an" object of the base class.

D. Inheritance means an object's behavior may change based on whether this object is created for the base class or a derived class at run-time.

E. In Java, polymorphism is achieved using abstract classes.

3 Object-Oriented Design and Programming

3.1 Abstract C++ Class

`Shape` is an abstract class and it contains a public function `area`; the function takes no parameter and returns `double`. The `area` function must be overridden by derived classes. Write the class declaration for `Shape`

Answer:

```
class Shape {
public:
virtual double area() = 0;
};
```

3.2 Prevent Inheritance in Java

What word in Java can be added in front of the word `class` so that the class cannot have any derived class.

Answer: final

3.3 Object Creation in C++

In C++, an object can be created in this way:

```
AClassName anobject(parameters for constructor);
```

How to create a C++ object using a pointer?

Answer: `AClassName * apointer = new AClassName(parameters for constructor);`

3.4 Copy Constructor in C++

Write the copy constructor for class `X`

Answer: `X (const X & xin)`

```
class X {
int p;
public:
   X( int q ) {
     p = q;
     cout << "X(int)" << endl;
   }
```

```
   // what should be here for copy constructor?
   {
     p = xin.p;
     cout << "X(const X&)" << endl;
   }

   int getp() const { return p; }
};
```

3.5 Destructor in C++

Write the appropriate destructor to prevent memory leak

Answer:

```
~X() {
delete [] p;
}
```

```
class X {
int * p;
int size;
public:
   X(int s) {
     size = s;
     p = new int[size];
   }
   // write the destructor
};
```

3.6 Java Set

Fill in the missing line (or lines).

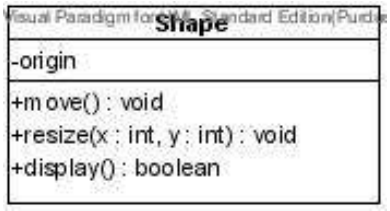
Answer: `Iterator iter = animals.iterator();`

```
import java.util.*;
class SetOps {
    public static void main( String[] args )
    {
        Set<String> animals = new TreeSet<String>();
        animals.add( "cheetah" );
        animals.add( "lion" );
        animals.add( "cat" );
        animals.add( "elephant" );
        animals.add( "cat" );
```

```
        System.out.println( animals );
        System.out.println( animals.size() );
        animals.remove( "lion" );
        System.out.println( animals );
        // missing line
        while ( iter.hasNext() ) {
            System.out.println( iter.next() );
        }
    }
}
```

3.7 UML Class Diagram

What is correct about this UML class diagram?



Answer: D

A. The three methods are protected.

B. `Shape` is an abstract class.

C. The method `move` returns false if the shape cannot move.

D. The class has an attribute called origin and it is private.

E. The method `display` takes one parameter and it is of type `boolean`.

3.8 UML Use Case

What is correct about this UML class diagram?

Answer: B

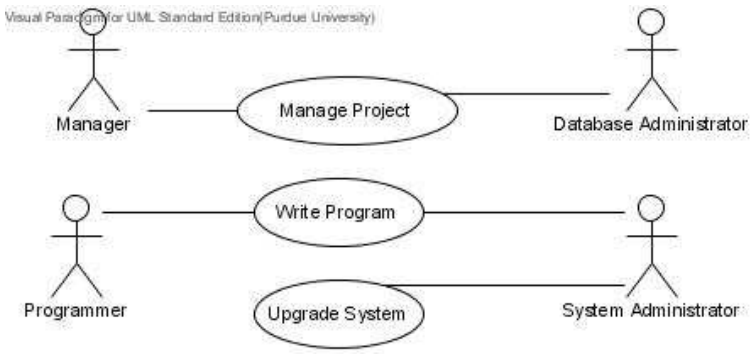
A. A use case specifies the interface of a class.

B. Two people may manage the project; one is the `Manager` and the other is the `Database Administrator`.

C. A use case shows the implementation of a class.

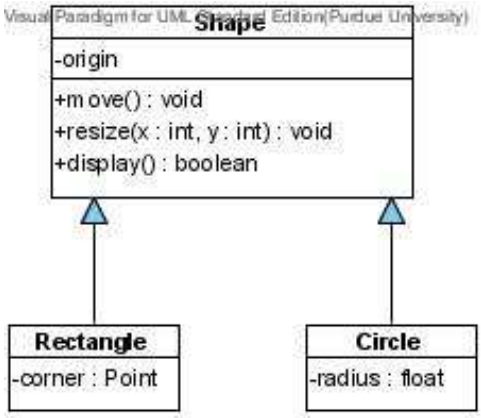
D. An actor can use only one case.

E. The system can be upgraded by both `Programmer` and `System Administrator`.



3.9 UML Class Diagram

What is correct about this UML class diagram?



Answer: E

A. `Rectangle` does not support the `move` method.

B. `Shape` is derived from `Rectangle` and `Circle`.

C. `Rectangle` must override the `display` method.

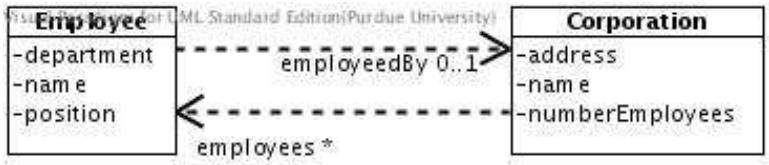
D. `Circle` has only one attribute called `radius`.

E. `Rectangle` does not have the `radius` attribute.

3.10 UML Relationship Diagram

What is correct about this UML relationship diagram?

Answer: C



A. `Employee` is a derived class of `Corporation`

B. `Employee` is the base class of `Corporation`

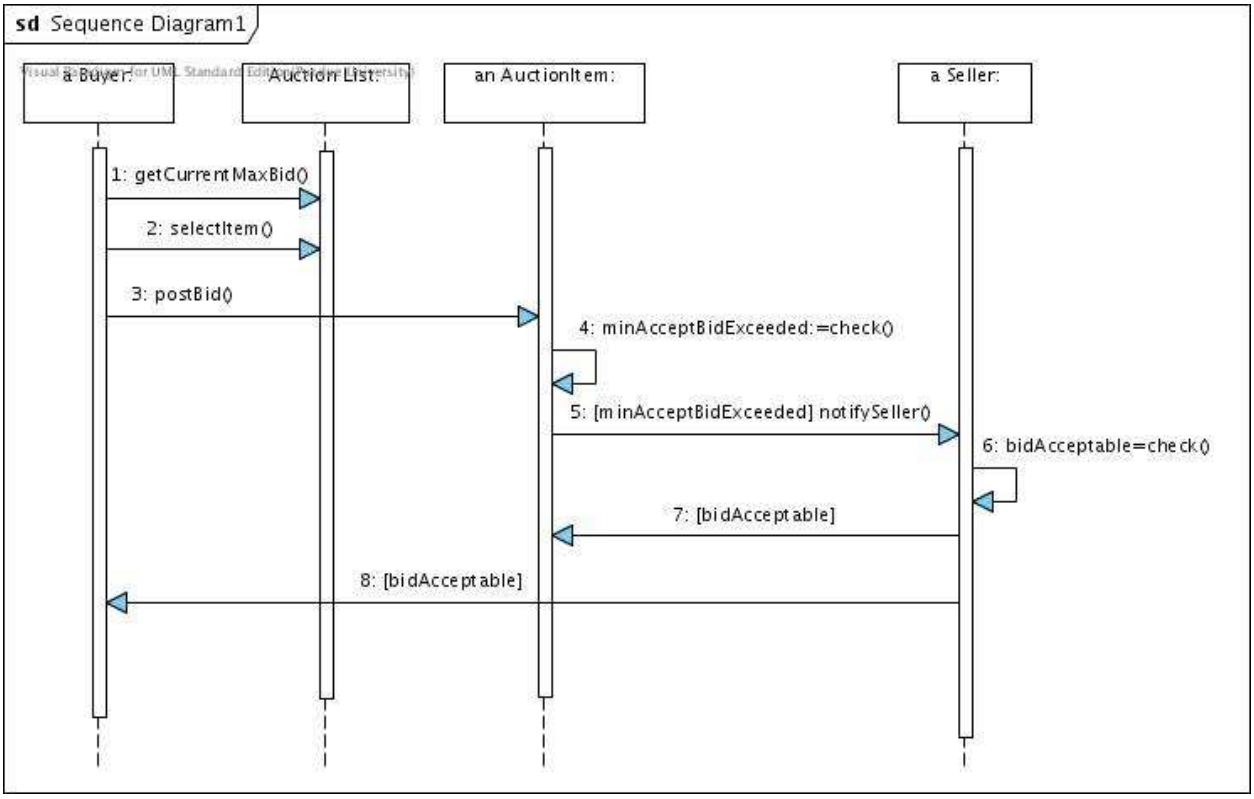
C. A `Corporation` object may have zero `Employee` object.

D. A `Corporation` object must have zero or one `Employee` object.

E. A `Corporation` object must be created before any `Employee` object.

3.11 UML Sequence Diagram

What is correct about this UML sequence diagram?



Answer: C

A. The `AuctionList` object is created after receiving the `getCurrentMaxBid()` message.

B. The `selectItem()` message may occur after the `postBid()` message.

C. The `notifySeller()` message is sent only if `minAcceptBidExceeded` condition is true.

D. In the third step, the `AuctionItem` object sends the `postBid()` message to the `Buyer` object.

E. The `Seller` object is created after sending the `bidAccepted()` message.

3.12 C++ Namespace

What is the output of this program?

Answer: 1:f 2:f 3:b

```
#include <iostream>
using namespace std;
namespace Module1 {
  class X {
  public:
    void foo(){ cout << "1:f "; }
    void bar(){ cout << "1:b "; }
  };
}
namespace Module2 {
  class X {
  public:
    void foo(){ cout << "2:f "; }
    void bar(){ cout << "2:b "; }
  };
}
namespace Module3 {
  using namespace Module1;
  class Z {
  public:
    void foo(){ cout << "3:f "; }
    void bar(){ cout << "3:b "; }
  };
}

int main()
{
  using namespace Module1;
  X xobj1;
  xobj1.foo();
  using namespace Module2;
  Module2::X xobj2;
  xobj2.foo();
  Module3::Z zobj3;
  zobj3.bar();
  return 0;
}
```

Contents