ECE 462 Final Exam

15:20-17:20, December 15, 2007

1 Overloading and Overriding

1.1 Overloading in Java

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

Answer: cannot compile

```
class Base {
    public void foo() {
        System.out.println( "Bf1" );
    }
    public void foo( int i ) {
        System.out.println( "Bf2" );
    }
}

class Derived extends Base {
    public void foo() {
        System.out.println( "Df1" );
    }
    public void foo( int i, int j ) {
        System.out.println( "Df2" );
    }
}

class J3Q1 {
    public static void main( String[] args )
    {
        Base d = new Derived();
        d.foo();
        d.foo( 3 );
        d.foo( 3, 4 );
    }
}
```

1.2 Overloading and Overriding and Class Hierarchy in C++

Which line of this program causes compile-time error? If there are multiple answers, you need to answer only one. If the program has no compile-time error, **write the program's output**.

Answer: Line 7 In member function 'void Base::bar(int)': 7 : error: no matching function for call to 'Base::foo(int&)'

```
 1  #include <iostream>
 2  using namespace std;
 3  class Base {
 4  public:
 5    Base() { }
 6    virtual void bar() { foo(); }
 7    void bar(int x) { foo(x); }
 8    virtual void foo() { cout <<"Bf1"<< endl;}
 9  };
10  class Derived : public Base {
11  public:
12    Derived() { }
13    ~Derived(){}
14    virtual void foo() { cout << "Df1" << endl; }
15    virtual void foo(int x) { cout <<"Df2"<< endl;}
16  };
17  int main() {
18    Derived* p = new Derived;
19    p->bar();
20    delete p;
21    return 0;
22  }
```

1.3 Overloading and Overriding

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

Answer: Df

```
class Base {
    public Base() {}
    public void foo(){
        System.out.println("Bf" );
    }
    public void bar() { foo(); }
}

class Derived extends Base {
    public void foo() {
        System.out.println( "Df" );
    }
    public Derived() {
    }
}

class J3Q3 {
    public static void main( String[] args ) {
        Base p = new Derived();
        p.bar();
    }
}
```

1.4 Overloading in C++ and Java

Which statement is correct?

Answer: A, B, C, or D (or any combination of them)

A. In C++, if a function is overloaded, it can still be overridden in a derived class.

B. In Java, if a function is overloaded, it can still be overridden in a derived class.

C. Overloaded functions in C++ can use primitive types or programmer-defined classes.

D. All above.

E. In Java, overloaded functions can **not** use objects as parameters.

1.5 Overloading in C++

Which statement is correct?

Answer: A

A. In C++, an integer **cannot** be promoted to string automatically.

B. In C++, only primitive types can distinguish which version of the overloaded function. Objects **cannot** decide the overloaded function.

C. In C++, a function can be overloaded only if it is a method in a class.

D. In C++, if a function is overloaded, a derived class must override all versions of the overloaded function.

E. In C++, a string argument may be automatically converted to double to match an overloaded function.

1.6 Overloading and Overriding

What is the output of this program? You can write the steps how `val` is changed without doing the calculation. For example, you can write `1 + 2` instead of `3`. If the program cannot compile, write "cannot compile".

Answer: 7 (= 4 + 3). The value is modified in this sequence: val = 1, val = 5, val += 6, val = 4, val += 3, val = 7

```
class MyBaseClass {
    protected int val;
    public MyBaseClass() { val = 1; }
    public void foo() { val += 2; }
    public void foo(int i) { val += 3; }
    public void foo(String str) { val = 4; }
    public int getVal() { return val; }
}

class MyDerivedClass extends MyBaseClass {
    public MyDerivedClass () { val = 5; }
    public void foo() { val += 6; }
}

class J3Q6 {
    public static void main(String[] args)
    {
        MyBaseClass mobj = new MyDerivedClass();
        String str = new String("hello");
        mobj.foo();
        mobj.foo(str);
        mobj.foo(4);
        System.out.println("val = " + mobj.getVal());
    }
}
```

2 Template Classes and the STL Library in C++

2.1 Template in C++

Which statement is correct?

Answer: A

A. A stack can be efficiently implemented using a list.

B. A list can be efficiently implemented using a vector.

C. A vector can be efficiently implemented using a queue.

D. A list can be efficiently implemented using a set.

E. A set can be efficiently implemented using a stack.

2.2 Container and Iterator

Replace /* here */ by declaring (and defining) an iterator.

Answer: ListIterator iter = animals.listIterator();

```
import java.util.*;
class J5Q5 {
    public static void main( String[] args )
    {
        List<String> animals = new ArrayList<String>();
        animals.add( "cheetah" );
        animals.add( "lion" );
        animals.add( "cat" );
        animals.add( "fox" );
        animals.add( "cat" );
        /* here */ /* ---------- */
        while ( iter.hasNext() ) {
            System.out.println( iter.next()  );
            /* output: cheetah lion cat fox cat */
        }
    }
}
```

2.3 C++ Set and Class Hierarchy

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

Answer: B0 D1 B2 (different orders acceptable, no duplicates)

```
#include <iostream>
#include <string>
#include <set>
using namespace std;

class BaseC
{
protected:
  int b_val;
public:
  BaseC(int val): b_val(val) {}
  virtual void print() { cout << "B" << b_val << endl; }
};

class DerivedC: public BaseC
{
public:
  DerivedC(int val): BaseC(val) {}
  virtual void print() { cout << "D" << b_val << endl; }
};
int main()
{
  set<BaseC*> bset;
  BaseC * bobj[3];
  bobj[0] = new BaseC(0);
  bobj[1] = new DerivedC(1);
  bobj[2] = new BaseC(2);
  bset.insert(bobj[0]);
  bset.insert(bobj[1]);
  bset.insert(bobj[2]);
  bset.insert(bobj[2]);
  bset.insert(bobj[1]);
  bset.insert(bobj[0]);
  typedef set<BaseC*>::const_iterator CI;
  for (CI iter = bset.begin();
       iter != bset.end();
       iter++)
    {
      (* iter) -> print();
    }
  return 0;
}
```

2.4 C++ Template

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

Answer: 15 h ece462 6.5

```
#include <iostream>
#include <string>
using namespace std;
template <class Txxx, class T2> class Container2
{
public:
  Container2(Txxx t1in, T2 t2in): c2_t1(t1in), c2_t2(t2in)
  { /* nothing */ }
  Txxx getT1(void) { return c2_t1; }
  T2 getT2(void) { return c2_t2; }
private:
  Txxx c2_t1;
  T2 c2_t2;
};
int main(void)
{
  Container2<int, char> obj1(15, 'h');
  cout << obj1.getT1() << " " << obj1.getT2() << endl;

  Container2<string, float> obj2("ece462", 6.5);
  cout << obj2.getT1() << " " << obj2.getT2() << endl;
  return 0;
}
```

2.5 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.

2.6 C++ List

What is the output of this program?

Answer: `5 c c g d f` *(The order must be correct.)*

```
#include <iostream>
#include <list>
#include <string>
using namespace std;
int main(void)
{
  list<string> charlist;
  list<string>::iterator iter;
  charlist.push_back("c");
  charlist.push_back("g");
  charlist.push_back("d");
  charlist.push_back("f");
  charlist.push_front("c"); // --- front, not back

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

  return 0;
}
```

3 Object-Oriented Programming using C++ and Java

3.1 C++ Function Parameter Passing

Which statement is correct?

Answer: E

```
/* C1 */
#include <iostream>
using namespace std;

void g( int y ) { y++; }
int main()
{
    int x = 100;
    g(x);
    cout << x << endl;
    return 0;
}
```

```
/* C2 */
#include <iostream>
using namespace std;
/* == notice the difference == */
void g( int & y ) { y++; }
int main()
{
    int x = 100;
    g(x);
    cout << x << endl;
    return 0;
}
```

A. The outputs of both programs are 100.

B. The outputs of both programs are 101.

C. The output of C1 is 101; the output of C2 is 100.

D. The output of C1 is 100; C2 does **not** compile.

E. The output of C1 is 100; the output of C2 is 101.

3.2 C++ and Java Function Parameter Passing

Which statement is correct?

Answer: C

```
/* Java */



class User {

    String name;
    int age;
    User(String nm, int a)
      { name = nm; age = a;}
}
class C9Q2 {
    static void swap(User s, User t) {
        User temp = s;
        s = t;
        t = temp;
    }
    public static void main(String[] args)
    {
        User u1 = new User("AAA", 95);
        User u2 = new User("BBB", 98);
        swap( u1, u2 );
        System.out.println( u1.name );
    }
}
```

```
/* C++ */
#include <iostream>
#include <string>
using namespace std;
class User {
public:
    string name;
    int age;
    User(string nm, int a)
      { name = nm; age = a;}
};

void swap(User s, User t) {
    User temp = s;
    s = t;
    t = temp;
}
int main()
{
    User u1("AAA", 95);
    User u2("BBB", 98);
    swap( u1, u2 );
    cout << u1.name  << endl;
    return 0;
}
```

A. Both Java and C++ output BBB.

B. Java outputs AAA; C++ outputs BBB.

C. Both Java and C++ output AAA.

D. Java outputs BBB; C++ outputs AAA.

E. Java outputs AAA; C++ does **not** compile.

3.3 Object Passing and Copy Constructor

What is the output of this program?

Answer: 3

```
#include <iostream>
#include <string>
using namespace std;
class User {
public:
    string name;
    int age;
    static int counter;
    User(string nam, int yy )
       { name = nam; age = yy; counter ++; }
    User(const User & orig)
       { name = orig.name; age = orig.age; counter ++; }
    int getCounter()
       { return counter; }
};
int User::counter = 0; // initialize static attribute
User f( User usr ) { return usr; }
int main()
{
    User u( "Xino", 120 );
    User y = f( u );
    cout << y.getCounter() << endl;
    return 0;
}
```

3.4 C++ and Java Function Parameter Passing

Which statement is correct?

Answer: E

```
/* Java */


class C9Q4 {
    static void swap(int x1, int x2) {

        int temp = x1;
        x1 = x2;
        x2 = x1;
    }
    public static void main(String[] args)
    {
        int u1 = 3;
        swap( u1, 5 );
        System.out.println( u1 );
    }
}
```

```
/* C++ */
#include <iostream>
#include <string>
using namespace std;
void swap(int & x1, int & x2) {
  /* notice & */
  int temp = x1;
  x1 = x2;
  x2 = x1;
}
int main(int argc, char * argv[])
{
  int u1 = 3;
  swap( u1, 5 );
  cout << u1 << endl;
  return 0;
}
```

A. Both Java and C++ output 3.

B. Java outputs 5; C++ outputs 3.

C. Java outputs 3; C++ outputs 5.

D. Both Java and C++ output 5.

E. Java outputs 3; C++ does **not** compile.

3.5 Primitive Types and Parameter Passing

What is the output of this program (in hexadecimal)?

Answer: 837e or 837E

```
class IntClass
{
    public IntClass(int u) { x = u; }
    public void print()
    { System.out.println(Integer.toHexString(x)); }
    public int x;
}

class C9Q5 {
    public static void main(String[] args)
    {
        IntClass u1 = new IntClass (0x124837F); // 0x means hexadecimal
        incr( u1 );
        u1.print();
    }
    static void incr(IntClass obj) {
        char y =  (char) obj.x;
        y --;
        obj.x = y;
    }
}
```

3.6 Object Passing and Copy Constructor

What is the output of this program?

Answer: 2

```
#include <iostream>
#include <string>
using namespace std;
class User {
public:
    string name;
    int age;
    static int counter;
    User(string nam, int yy )
       { name = nam; age = yy; counter ++; }
    User(const User & orig)
       { name = orig.name; age = orig.age; counter ++; }
    int getCounter()
       { return counter; }
};
int User::counter = 0;
User & f( User & usr ) { return usr; } // notice ‘‘&’’
int main()
{
    User u( "Xino", 120 );
    User y = f( u );
    cout << y.getCounter() << endl;
    return 0;
}
```

3.7 Object Creation

Which statement is correct?

Answer: D

```
/* C++ */
#include <iostream>
using namespace std;
class CX {
public:
  CX(int s) {
    cx_size = s;
    // --- ATTENTION --- notice ``int *''
    int * cx_data = new int[s];
    for (int iter = 0; iter < s;
         iter ++) {
      cx_data[iter] = iter;
    }
  }
  virtual void print() {
    for (int iter = 0; iter < cx_size;
         iter ++) {
      cout << cx_data[iter] << " ";
    }
    cout << endl;
  }
  virtual ~ CX() {
    delete [] cx_data;
  }
private:
  int cx_size;
  int * cx_data;
};

int main(int argc, char * argv[])
{
  if (argc > 1) {
    CX cobj(4);
    cobj.print();
  } else {
    CX cobj(8);
    cobj.print();
  }
  return 0;
}
```

```
/* Java */


class CX {
  public CX(int s) {

    cx_size = s;

    cx_data = new int[s];
    for (int iter = 0; iter < s;
         iter ++) {
      cx_data[iter] = iter;
    }
  }
  public void print() {
    for (int iter = 0; iter < cx_size;
          iter ++) {
      System.out.print(cx_data[iter]);
      System.out.print(" ");
    }
    System.out.println("");
  }

  private int cx_size;
  private int [] cx_data;
}


class C15Q1 {
  public static void main(String []
                          args ) {
    if (args.length > 1) {
      CX cobj = new CX(4);
      cobj.print();
    } else {
      CX cobj = new CX(8);
      cobj.print();
    }
  }
}
```

A. C++ does **not** compile (i.e. syntax error) but Java can compile.

B. C++ can compile but Java does **not** compile.

C. Both can compile and both can execute. If no run-time parameter is given, both prints `0 1 2 3 4 5 6 7`.

D. C++ has a (or multiple) run-time error but Java has no run-time error.

E. C++ has no run-time error but Java has a (or multiple) run-time error.

3.8 Virtual and Non-Virtual Functions in C++

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

Answer: `BNV DV`

```
#include <iostream>
using namespace std;
class BaseC {
public:
  virtual void vfunc() {
    cout << "BV" << endl;
  }
  void nvfunc() {
    cout << "BNV" << endl;
    vfunc();
  }
  virtual void vfunc2() {
    cout << "BV2" << endl;
    nvfunc();
  }
};

class DerivedC: public BaseC {
public:
  virtual void vfunc() {
    cout << "DV" << endl;
  }
  void nvfunc() {
    cout << "DNV" << endl;
    vfunc();
  }
  virtual void vfunc2() {
    cout << "DV2" << endl;
    nvfunc();
  }
};

int main(int argc, char * argv[]) {
  BaseC * bobj = new DerivedC;
  bobj -> nvfunc();
  return 0;
}
```

3.9 Multiple Virtual Inheritance in C++

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

Answer: `31 2 3 4`

```
#include <iostream>
using namespace std;
class X {
protected:
  int xval;
public:
  X (int xin) : xval(xin) {}
};

class Y : virtual public X {
protected:
  int yval;
public:
  Y (int xin, int yin) : X(xin), yval(yin) {}
};

class Z : virtual public X {
protected:
  int zval;
public:
  Z (int xin, int zin) : X(xin), zval(zin) {}
};

class U : public Y, public Z{
  int uval;
public:
  U (int xin, int yin, int zin, int uin )
    : X(xin + 30), Y(xin + 20, yin), Z(xin + 10, zin), uval(uin) {}
  void print() {
    cout << xval << " " << yval << " " << zval << " " << uval << endl;
  }
};

int main()
{
  U uobj(1, 2, 3, 4);
  uobj.print();
  return 0;
}
```

3.10 Multiple Inheritance in C++

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

Answer: cannot compile

error: type 'X' is not a direct base of 'U'

```
#include <iostream>
using namespace std;
class X {
protected:
  int xval;
public:
  X (int xin) : xval(xin) {}
};

class Y : public X { // **** there is no ``virtual'' ****
protected:
  int yval;
public:
  Y (int xin, int yin) : X(xin), yval(yin) {}
};

class Z : public X {
protected:
  int zval;
public:
  Z (int xin, int zin) : X(xin), zval(zin) {}
};

class U : public Y, public Z{
  int uval;
public:
  U (int xin, int yin, int zin, int uin )
    : X(xin + 30), Y(xin + 20, yin), Z(xin + 10, zin), uval(uin) {}
  void print() {
    cout << xval << " " << yval << " " << zval << " " << uval << endl;
  }
};

int main()
{
  U uobj(1, 2, 3, 4);
  uobj.print();
  return 0;
}
```

3.11 TCP Port

What is the default port number for HTTP server?

Answer: 80

3.12 Java Chat Server

Which statement is correct?

Answer: E

A. The server creates a port by calling `new TcpSocket()`.

B. After creating a socket, the server calls `listen` and wait for a client's connection.

C. A server can respond to only one client at any moment. If another client wants to connect to the server, this client has to wait until the currently connected client disconnect.

D. The client and the server must **not** be on the same machine (with the same IP address).

E. The server can use an `InputStream` object to read from the socket.

3.13 Qt (4.3) Chat Server

Which statement is correct about Qt (4.3) Chat Server?

Answer: A

A. The server creates a port by calling `new QTcpServer()`.

B. After creating a socket, the server calls `accept` and wait for a client's connection.

C. The server must use multiple threads, by creating objects from a class that is derived from `QThread`, to respond to multiple clients simultaneously. If the server does **not** create such objects, only one client can connect to the server at any moment.

D. When a client is connected, `SIGNAL( readyRead() )` is issued.

E. Creating a server port `new QTcpPort()` can give a preferred port number, such as `new QTcpPort(5432)`.

3.14 Java Chat Server

Which statement is correct?

Answer: C

```
import java.io.*;
import java.net.*;
import java.util.*;

public class ChatServer {
    public static List<ClientHandler> clientList =
        new ArrayList<ClientHandler>();

    public static void main( String[] args ) {
        try {
            ServerSocket server = new ServerSocket( 5000 );
            for (;;) {
                Socket socket = server.accept();
                System.out.print( "A new client checked in:   " );
                ClientHandler clh = new ClientHandler( socket );
                clientList.add( clh );
                clh.start();
            }
        } catch( Exception e ) { System.out.println( e ); }
    }
}

/////////////////  class ClientHandler extends Thread  ////////////////
class ClientHandler extends Thread {
    private String userName;
    private Socket sock;
    private static List<String> chatStore = new ArrayList<String>();
    private BufferedReader buff_reader = null;
    private PrintWriter out = null;

    public ClientHandler( Socket s ) {
        try {
            sock = s;
            out = new PrintWriter( sock.getOutputStream() );
            InputStream in_stream = sock.getInputStream();
            InputStreamReader in_reader =
                new InputStreamReader( in_stream );
            buff_reader = new BufferedReader( in_reader );

            // ask for user name
            out.println( "\n\nWelcome to Avi Kak's chatroom");
            out.println();
            out.println(
                        "Type \"bye\" in a new line to terminate session.\n" );
            out.print( "Please enter your first name: " );
```

```
        out.flush();
        userName = buff_reader.readLine();
        out.print("\n\n");
        out.flush();
        System.out.print( userName + "\n\n" );

        // show to new client all the chat
        // that has taken place so far
        if ( chatStore.size() != 0 ) {
            out.println( "Chat history:\n\n" );

            ListIterator iter = chatStore.listIterator();
            while ( iter.hasNext() ) {
                out.println( (String) iter.next() );
            }
            out.print("\n\n");
            out.flush();
        }
    } catch( Exception e ) {}
}

public void run() {
    try {
        boolean done = false;
        while ( !done ) {
            out.print( userName + ": " );
            out.flush();
            String str = buff_reader.readLine();

            if ( str.equals( "bye" ) ) {
                str = userName + " signed off";
                done = true;
            }
            String strWithName = userName + ": " + str;
            chatStore.add( strWithName );
            ListIterator iter =
                ChatServer.clientList.listIterator();
            while ( iter.hasNext() ) {
                ClientHandler cl = (ClientHandler) iter.next();
                if ( this != cl ) {
                    cl.out.println();
                    cl.out.println( strWithName );
                    cl.out.print( cl.userName + ": " );
                    cl.out.flush();
                }
            }
        }
        System.out.println( userName + " signed off" + "\n\n" );
        buff_reader.close();
        out.close();
        sock.close();
```

```
        } catch ( Exception e ) {}
    }
}
```

A. Calling `ServerSocket` creates a socket that connects to a server.

B. After creating a socket, the server calls `listen` and wait for a client's connection.

C. The server does **not** have to use multiple threads but can still broadcast to multiple clients.

D. When a client is connected, `SIGNAL( readyRead() )` is issued.

E. A server creates a socket for clients' connections by calling `new Socket( IPAddress, 80 );`

3.15 Qt (4.3) Chat Server

Please fill the code for `ClientHandler::broadcastClient(QString message)`.

Answer:

```
ClientHandler * handler;

for (unsigned int index = 0; index < clientVector.size(); index ++)

{

handler = clientVector[index];

*(handler-> os) << message;

(handler-> os) -> flush();

}
```

```
// ChatServer.h
// This program has memory leak.  Do not worry about it.
#ifndef CHATSERVER_H
#define CHATSERVER_H
#include <QtNetwork>
#include <QString>
#include <QThread>
#include <QApplication>
#include <vector>
using namespace std;

class ClientHandler : public QObject
{
  Q_OBJECT

private:
  QString chatName; // assume names of the clients are unique
  QTcpSocket* handlerSocket;
  QTextStream* os;
  static QList<QString*> * chatStore;
  void broadcastClient(QString);

public:
  ClientHandler( QTcpSocket* sock);
  virtual ~ClientHandler();
  static vector<ClientHandler *> clientVector;

private slots:
  void readFromClient();
};

class ChatServer: public QObject {
  Q_OBJECT
```

```
private:
  QTcpServer * server;

public:
  ChatServer( );
  ChatServer(const ChatServer & cs);
  virtual ~ChatServer();

public slots:
  void connectNewClient( );
};
#endif


//ChatServer.cc

#include "ChatServer.h"
#include <iostream>
using namespace std;

// initialize static members
QList<QString*> * ClientHandler::chatStore = new QList<QString*>;
vector<ClientHandler *> ClientHandler::clientVector;

ChatServer::ChatServer( )
{
  server = new QTcpServer();
  if (! server->listen())
    {
      qWarning( "Failed to register the server port" );
      exit( 1 );
    }
  cout << "Server port " << server->serverPort() << endl;
  connect(server, SIGNAL(newConnection()), this, SLOT(connectNewClient()));
}

void ChatServer::connectNewClient()
{
  QTcpSocket* socket = server->nextPendingConnection();
  // socket->setSocket( socketFD );
  ClientHandler* clh = new ClientHandler( socket );
  ClientHandler::clientVector.push_back(clh);
  cout << "A new client connected " << endl;
}

ChatServer::~ChatServer(){}
ClientHandler::ClientHandler( QTcpSocket* socket)
  : chatName(""), handlerSocket( socket )
{
  os = new QTextStream( handlerSocket );
  (*os) << "Welcome to a chat room powered by C++\n";
```

```
  (*os) << ">>>>    Enter 'bye' to exit   <<<\n";
  (*os) << "Enter chat name: ";
  os -> flush();
  connect( handlerSocket, SIGNAL( readyRead() ),
           this, SLOT( readFromClient() ) );
}

ClientHandler::~ClientHandler(){}

void ClientHandler::broadcastClient(QString message)
{
  // ******************
  // ---- FILL HERE ---
}

void ClientHandler::readFromClient() {
  QTcpSocket* sock = (QTcpSocket*) sender();
  while ( sock->canReadLine() ) {
    QString qstr = sock->readLine();
    qstr = qstr.trimmed(); // remove white space
    if ( chatName == "" )
      {
        chatName = qstr;
        QString outgoing = "\nMessage from chat server: " +
          chatName + " signed in\n";
        broadcastClient(outgoing);
      }
    else if ( qstr == "bye" )
      {
        QString outgoing = "\nMessage from the chat server: " +
          chatName + " signed off\n";
        broadcastClient(outgoing);
        handlerSocket->close();
        handlerSocket = 0;
      }
    else
      {
        QString outgoing = "\n" + chatName + ": " + qstr + "\n";
        broadcastClient(outgoing);
      }

    // A chatter's terminal always shows his/her own name at beginning
    // of a new line.
    for (unsigned int index = 0; index < clientVector.size(); index ++)
      {
        ClientHandler * handler = clientVector[index];
        *(handler-> os) << (handler->chatName + ": ");
        (handler-> os) -> flush();
      }
  }
}
```

```
int main( int argc, char* argv[] )
{
  QApplication app( argc, argv );
  ChatServer* server = new ChatServer( );
  return app.exec();
}
```

3.16 Operator Overloading and Object-Oriented Programming

Which statement is correct?

Answer: B

A. User-defined operator overloading is supported in C++ and Java.

B. If operators < and == are overloaded, calling <= will **not** be automatically converted to calling < and ==.

C. A binary operator can be implemented as a member function, but a unary operator **cannot** be a member function.

D. In C++, an operator can be overloaded for user-defined objects only. In Java, an operator can be overloaded for primitive types only.

E. User-defined operator overloading is necessary for polymorphism.

3.17 Exception in Java

Which statement is correct?

Answer: A

```
class MyException /* A */
{
    String message;
    public MyException( String mess ) { message = mess; }
}

class Test
{
    static void f() /* B */
    {
        throw new MyException( "Hello from f()" );
    }
    public static void main( String[] args )
    {
        /* C */
```

```
        try {
            f();
        } catch( MyException e ) { /* D */
            System.out.println( e.message );
        }
    }
}
```

A. /* A */ must be replaced by `extends Exception`.

B. /* A */ can be replaced by `extends Exception` **or** left empty; either can work.

C. /* B */ can be replaced by `throws String;`.

D. /* C */ can call `f();`

E. /* D */ `try` does **not** have to be followed by `catch`, in the same ways as `if` does not have to be followed by `else`.

3.18 Operator Overloading as Member Functions

Which statement is correct?

Answer: C

```
#include <iostream>
using namespace std;
class MyComplex {
  double re, im;
public:
  MyComplex( double r, double i = 0) : re(r), im(i) {}
  MyComplex operator+( MyComplex) const;
  MyComplex operator-( MyComplex) const;
  double getReal() const { return re; }
  double getImag() const { return im; }
  friend ostream& operator<< ( ostream&, const MyComplex& );
};

MyComplex MyComplex::operator+( const MyComplex arg ) const {
  double d1 = re + arg.re;
  double d2 = im + arg.im;
  return MyComplex( d1, d2 );
}

MyComplex operator+(const double x, const MyComplex arg ) {
  double d1 = x + arg.getReal();
  double d2 = arg.getImag();
  return MyComplex( d1, d2 );
}

MyComplex MyComplex::operator-( const MyComplex arg ) const {
  double d1 = re - arg.re;
  double d2 = im - arg.im;
  return MyComplex( d1, d2 );
}

ostream& operator<< ( ostream& os, const MyComplex& c ) {
  os << "(" << c.re << ", " << c.im << ")" << endl;
  return os;
}

int main()
{
  MyComplex a(3, 4);
  MyComplex b(2, 9);
  MyComplex c = 4.7 + a; // <--- x
  MyComplex d = b + 9.2; // <--- y
  cout << c << endl;
  cout << d << endl;
  return 0;
```

```
}
```

A. Line x causes compile-time error.

B. Line y causes compile-time error.

C. There is no compile-time error in the program.

D. Both x and y cause compile-time error.

E. The program has compile-time error (one or multiple) but it is **not** caused by x or y.

3.19 Inheritance, Interface, and Implementation

Answer: A

Which statement is correct?

A. An abstract C++ class must have at least one method that is pure virtual.

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

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

D. If a C++ class is abstract, all methods must be pure virtual.

E. If class X is derived from class Y (`class X extends Y`), class Y is more specific than X. Y has all the attributes in X and probably some additional attributes.

3.20 Inheritance in Java

In Java, `Exception` is a class. How do you create a class that has the properties of this class?

Answer: `class X extends Exception`

3.21 Exception in C++

Which statement is correct?

Answer: B

```
#include <iostream>
#include <cstdlib>
using namespace std;
class Err {};
void f( int ); /* A */
int main()
{
  try
    {
      f(0); /* B */
    }
  catch ( Err )
    {
      cout << "caught Err" << endl;
      exit(0);
    }
  return 0;
}
void f(int j) /* C */
{
  cout << "function f invoked with j = " << j << endl;
  if (j == 3) throw Err();
  f( ++j ); /* D */
}
```

A. /* A */ must be replaced by `throw(int);`.

B. If /* B */ `f(0);` is replaced by `f(5);` no exception will be thrown but the program will eventually stack overflow.

C. /* A */ must be replaced by `throw(Err);`.

D. /* C */ must be replaced by `throw(Err);`.

E. /* D */ must put `f(++j);` inside a `try-catch` block.

3.22 Synchronization in Java

Which statement is correct about this program?

```
import java.util.*;
class SyncAlways extends Thread
{
    public static long total = 0;
    public static long NUMBER_ITERATION;
    synchronized public void addTotal() { total ++; }
    public void run()
    {
        for (long iter = 0; iter < NUMBER_ITERATION; iter ++)
            { addTotal(); }
    }
    public SyncAlways(long ITERATION)
    { NUMBER_ITERATION = ITERATION;  }
    public String toString()
    { return new String("Total = " + total); }
}

class SyncLast extends Thread
{
    public static long total = 0;
    public static long NUMBER_ITERATION;
    private long subtotal = 0;
    synchronized public static void addTotal(long sub)
    { total += sub; }
    public void run()
    {
        for (long iter = 0; iter < NUMBER_ITERATION; iter ++)
            { subtotal ++;  }
        addTotal(subtotal);
    }
    public SyncLast(long ITERATION)
    { NUMBER_ITERATION = ITERATION;  }
    public String toString()
    { return new String("Total = " + total); }

}

class C18Q1
{
    public static void main( String[] args )
    {
        long NUMBER_ITERATION = 1000000;
        int NUMBER_THREAD = 8;
        System.out.println("Iteration = " + NUMBER_ITERATION / 100000 +
                           " (millions) " + "Thread = " +
                           NUMBER_THREAD);
        SyncLast [] slt = new SyncLast[NUMBER_THREAD];
```

```
        for (int tcnt = 0; tcnt < NUMBER_THREAD; tcnt ++)
            { slt[tcnt] = new SyncLast(NUMBER_ITERATION); }
        try {
            for (int tcnt = 0; tcnt < NUMBER_THREAD; tcnt ++)
                { slt[tcnt].start(); }

            for (int tcnt = 0; tcnt < NUMBER_THREAD; tcnt ++)
                { slt[tcnt].join(); }
        } catch (InterruptedException ie) {
            System.out.println("exception caught");
        }

        SyncAlways [] saws = new SyncAlways[NUMBER_THREAD];
        for (int tcnt = 0; tcnt < NUMBER_THREAD; tcnt ++)
            { saws[tcnt] = new SyncAlways(NUMBER_ITERATION); }
        try {
            for (int tcnt = 0; tcnt < NUMBER_THREAD; tcnt ++)
                { saws[tcnt].start(); }

            for (int tcnt = 0; tcnt < NUMBER_THREAD; tcnt ++)
                { saws[tcnt].join(); }
        } catch (InterruptedException ie) {
            System.out.println("exception caught");
        }
        System.out.println("SyncLast   " + slt[0]);
        System.out.println("SyncAlways " + saws[0]);
    }
}
```

Answer: B

A. SyncLast and SyncAlways both *always* output 8000000.

B. SyncLast is *more likely* to output 8000000 than SyncAlways but neither is guaranteed to always output 8000000.

C. SyncAlways is *more likely* to output 8000000 than SyncLast but neither is guaranteed to always output 8000000.

D. It is **impossible** for SyncLast nor SyncAlways to produce 8000000.

E. The program has compile-time error.

3.23 Inheritance and Methods in Java

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

Answer: `B1 D2 B1 D2`

```
class BaseC {
  public void func1() {
  System.out.println("B1");
    func2();
  }
  public void func2() {
    System.out.println("B2");
  }
}

class DerivedC extends BaseC {
  public void func2() {
    System.out.println("D2");
  }
}

class O4 {
  public static void main(String [] args ) {
    BaseC bobj = new DerivedC();
    bobj.func1();
    DerivedC dobj = new DerivedC();
    dobj.func1();
  }
}
```

3.24 Java Applet

The following is an HTML page that loads an applet called `ball.class` and sets the width and the height to be 400. What should be put in `REPLACE THIS`?

Answer:

<applet code="ball.class" height="400" width="400">

```
<html>
<head>
<title>Bouncing Ball</title>
</head>
<body>
<h1>Bouncing Ball</h1>
REPLACE THIS
</applet>
</body>
</html>
```

3.25 Program Test Coverage

Which statement is correct?

Answer: B

A. If a program's test coverage is 100%, all possible execution paths have been tested.

B. If a program has *dead code*, the program's test coverage **cannot** reach 100%.

C. Encapsulation helps testing because all attributes are public.

D. Object-oriented programs should always use top-down testing.

E. If a line of statement does not execute in a test, this statement is called *dead code*.

3.26 Programming Assignment 2 (C++)

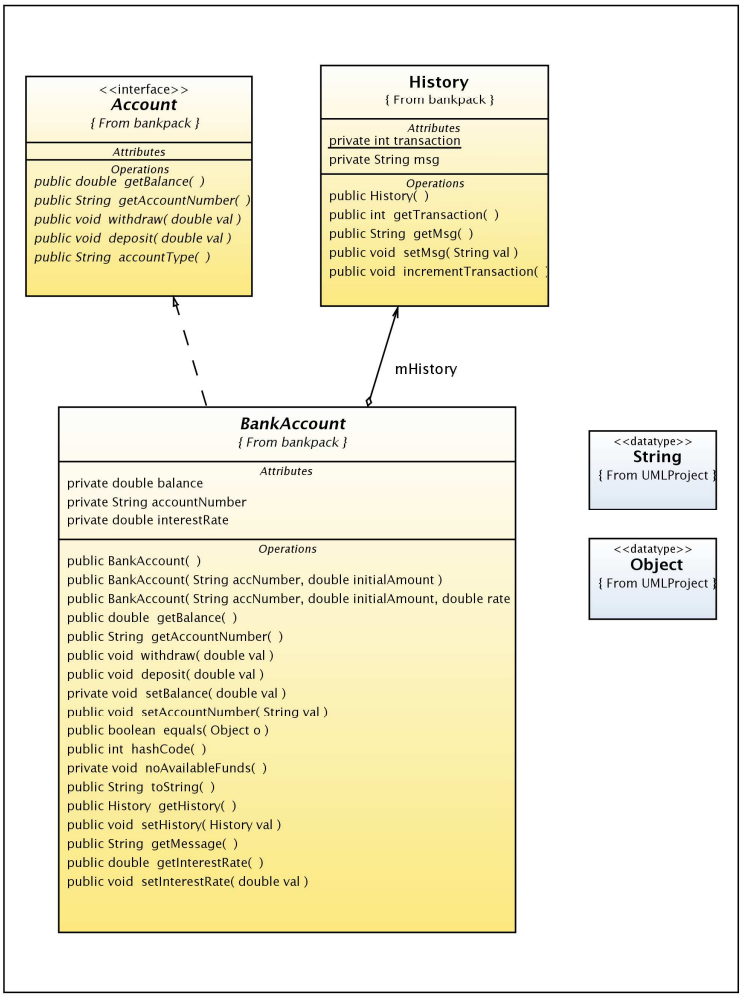
In Tetrix, after a `QPushButton` is created, how to instruct the program that function `buttonPressed` should be called when this button is clicked?

Answer: `connect(startButton, SIGNAL(clicked()), this, SLOT(buttonPressed()));`

The words `SIGNAL` *and* `SLOT` *must appear. The third parameter can be* `board`.

```
 startButton = new QPushButton(Start);
 /* fill this line */
 .... /* some more code */
void TetrixBoard::buttonPressed()
{
}
```

3.27 UML Diagrams: Class Relationship (Bank Account)



BankAccountDependencies

Which statement is correct? This question uses a diagram from "NetBeans 5.5 and 5.5.1 UML Modeling Documentation" (`http://www.netbeans.org/kb/55/uml-index.html`).

A larger print of the figure is available at the end of this exam.

Answer: E

A. `BankAccount` is a derived class from `History` (`BankAccount extends History`).

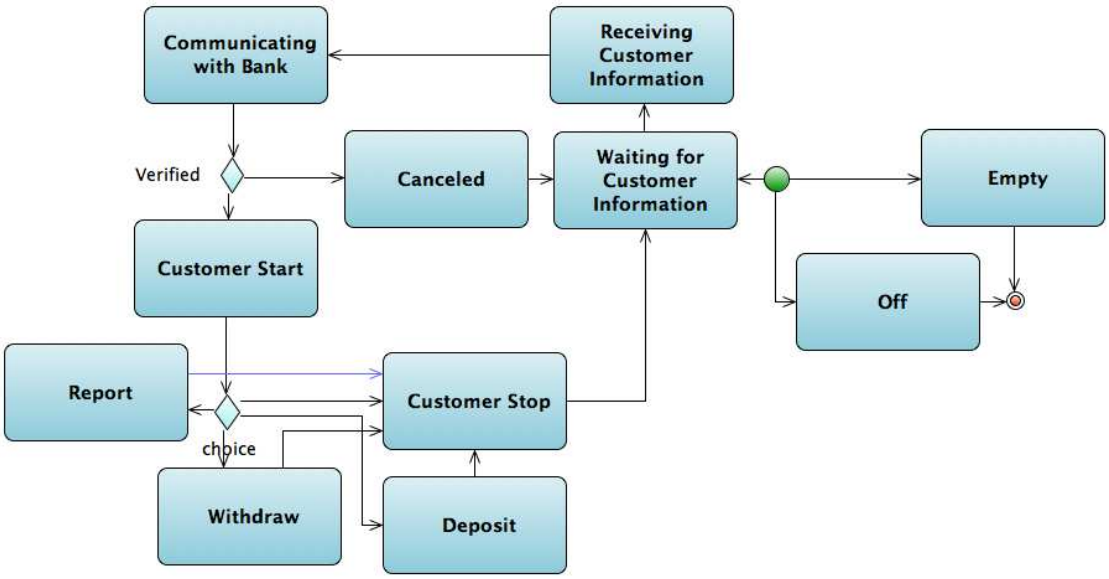
B. Calling `getAccountNumber` returns a 10-digit number.

C. The two methods `withdraw` and `deposit` are synchronized.

D. The method `setInterestRate` returns `false` if the input parameter is negative.

E. A `BankAccount` object can be created without an account number.

3.28 UML Diagrams: State Diagram (Bank Transaction)



Which statement is correct?

Answer: A

A. After a customer is verified, the customer can choose to receive a report, withdraw, or deposit.

B. After a transaction is canceled, the system enters the final state and no additional transaction is allowed.

C. A customer must choose deposit or withdrawal before providing the customer's information.

D. A customer can enter PIN (personal identification number) at most three times.

E. Once a customer's information is verified, the customer must deposit, withdraw, or request a report. There is no option to stop the transaction.

Contents