20th CSEC – Past Year Paper Solution *2016-2017 Sem 2*
CE/CZ 2002 – Object Oriented Design & Programming

1) a) The ways in which the this keyword can be used in Java is:
- It can be used to invoke the current class variables and member functions using the dot operator.
- this() can be used to invoke the class constructor.
- this can be passed as an argument to a method or a constructor call.
- this can be returned from a member function.

The ways in which the keyword super can be used in Java is:
- super can be used to refer to the immediate parent class instance variable.
- super can be used to invoke the immediate parent class method.
- super() can be used to invoke the immediate parent class constructor.

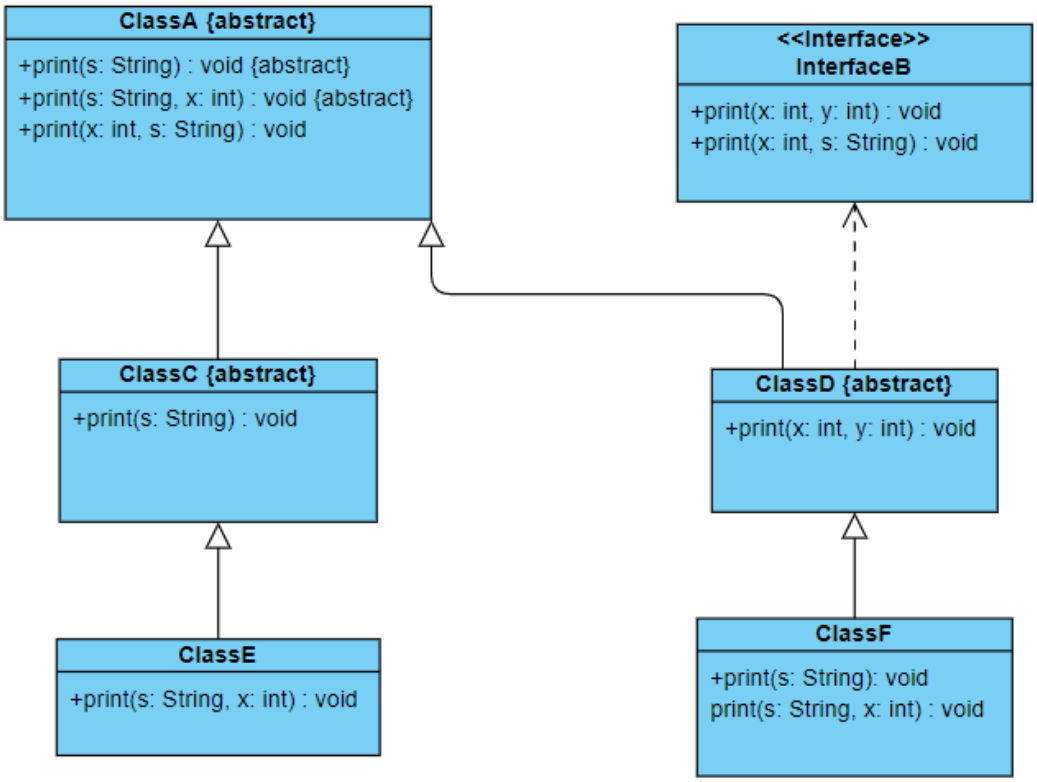
The keyword this is mainly used to refer to the current class while super refers to the immediate parent class.

b) Classes inside the package and subclasses inside or outside the package. The symbol used in UML for protected is '#'

c) Binding is the mechanism of creating a link between the method call and the actual method implementation. The two ways of binding are static binding and dynamic binding.
- Static binding uses type information for binding while dynamic binding uses objects to resolve binding.
- Overloaded methods are resolved using static binding while overridden methods are resolved using dynamic binding.

d) Encapsulation builds the barrier to protect an object's private data. Access to the private data can be done through public methods of the object class. Information hiding hides the implementation details of the class from the users.
It is implemented in java using private variables and public accessor and mutator functions.

2 a)



b)

```
public abstract class ClassA {
	public abstract void print (String s);
	public abstract void print (String s, int x);
	public void print (int x, String s) {
		System.out.println(x);
		System.out.println(s);
	}
}

public abstract class ClassC extends ClassA{
	public void print(String s) {
		System.out.println(s);
	}
}

public abstract class ClassD extends ClassA implements ClassB{
	public void print (int x, int y) {
		System.out.println(x);
		System.out.println(y);
	}
}

public class ClassE extends ClassC{
	public void print(String s, int x) {
		super.print(s);
		System.out.println(x);
```

```
        }
    }

    public class ClassF extends ClassD {
        public void print(String s) {
            System.out.println(s);
        }
        public void print(String s, int x) {
            System.out.println(s);
            System.out.println(x);
        }
    }
```

c) Both the function calls will result in the calling of the function in the ClassF due to dynamic binding of the function.

3) a)

```
//in receiver.h

class Receiver {
  private:
    string msg;
  public:
    virtual void receive(string msg);
    string getMsg();
    void setMsg(string msg);
}
```

b)

```
Header file: relay.h
#include <receiver.h>
#include <transmitter.h>

class Relay: public Receiver, public Transmitter {
   private:
       string advert;
       string refine(string msg);
   public:
       Relay(string advert);
       void receive(string msg);
}

Implementation file: relay.c
#include <relay.h>
// I am passing the message as a parameter as it needs to be
concatenated
string Relay :: refine(string msg) {
    return advert + " >> " + msg;
}

Relay :: Relay (string advert) {
    this.advert = advert;
```

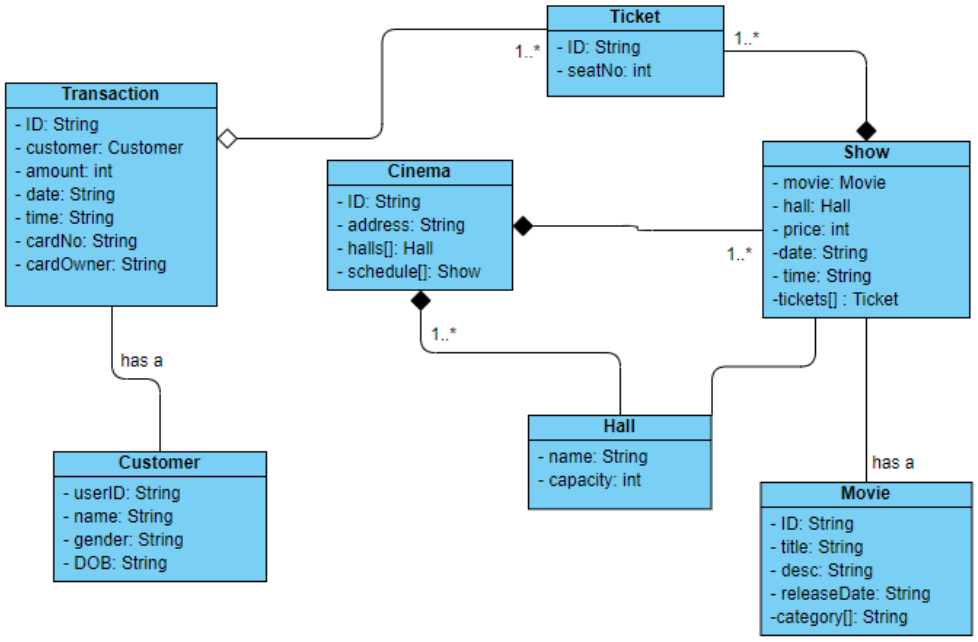
```
}

Void Relay :: receive (string msg) {
    Receiver.receive(msg);
    send ( refine(msg));
}
```

c) i) For this we can utlize the Open – Close principle. If we are able to create another class, say, SMSandEmail that inherits from the News class, we can have the client class utilize the new class instead of the News class. In this way changes will only have to be made to the Client class.

```
ii) void main() {
 SMSandEmail newChannel;
News A = newChannel;
Receiver B;
A.addReceiver(B);
//The notify method of the new class can implement the email
requirement
A.notify("test");
}
```

4) a)



b)

```
import java.util.ArrayList;

public class Member {

      public String name;
      public String ID;
      private int vote;

      public int initialVote(ArrayList<Member> MemberList) {
            return pollMembers(MemberList);
```

```
	}
	private int pollMembers(ArrayList<Member> MemberList) {
		int votes = 0;
		for (int i = 0 ; i < MemberList.size(); i++) {
			Member other = MemberList.get(i);
			if (this != other) {
				votes += other.getVote();
				displayMemberInfo(other);
			}
		}
		return votes;
	}

	public void displayMemberInfo(Member mem) {
		System.out.println(mem.ID + ": " + mem.name);
	}
	public int getVote() {
		 Return vote;
	 }
	public void setVote(int vote) {
		this.vote = vote;
	}
}
```

--End of Answers--