IRC_SKCT_Java2_MCQ_Inheritance_Polymomrphism

Test Summary

- No. of Sections: 1
- No. of Questions: 20
- Total Duration: 30 min

Section 1 - MCQ

Section Summary

- No. of Questions: 20
- Duration: 30 min

Additional Instructions:

None

Q1. Which of the following advantages do we lose by using multiple inheritance?

- Dynamic binding
- Polymorphism
- All of the listed options
- None of the listed options

Q2. Which symbol is used to create multiple inheritance?

- Dot
- Comma
- Dollar
- None of the mentioned

Q3. What is the syntax of inheritance of class?

- class name
- class name : access specifer
- class name : access specifer class name
- None of the mentioned

Q4. What is the output of this program?

```
#include <iostream>
using namespace std;
class BaseClass
```

```
{
int i;
public:
void setInt(int n);
int getInt( );
};
class DerivedClass : public BaseClass
{
int j;
public:
void setJ(int n);
int mul( );
};
void BaseClass::setInt(int n)
{
i = n;
}
int BaseClass::getInt( )
{
return i;
}
void DerivedClass::setJ(int n)
{
j = n;
}
int DerivedClass::mul( )
{
return j * getInt( );
}
int main( )
{
DerivedClass ob;
ob.setInt(10);
ob.setJ(4);
cout << ob.mul( );
return 0;
}
```

- 10
- 4
- 40
- None of the listed options

Q5. What is the output of this program?

```
#include <typeinfo>
#include <iostream>
using namespace std;
class Myshape
{
public:
virtual void myvirtualfunc( ) const {}
};
class mytriangle: public Myshape
{
public:
virtual void myvirtualfunc( ) const
{
};
};
int main( )
{
Myshape Myshape_instance;
Myshape &ref_Myshape = Myshape_instance;
try
{
mytriangle &ref_mytriangle = dynamic_cast<mytriangle&>(ref_Myshape);
}
catch (bad_cast)
{
cout << "Can't do the dynamic_cast lor!!!" << endl;
cout << "Caught: bad_cast exception. Myshape is not mytriangle.\n";
```

```
}
return 0;
}
```

(A) Can’t do the dynamic_cast lor!!!

(B) Caught: bad_cast exception. Myshape is not mytriangle.

(C) both A & B

(D) none of the mentioned

Q6. What is the output of this program?

```
#include <iostream>
using namespace std;
class Base1
{
protected:
int SampleDataOne;
public:
Base1( )
{
SampleDataOne = 100;
}
~Base1( )
{
}
int SampleFunctOne( )
{
return SampleDataOne;
}
};
class Base2
{
protected:
int SampleDataTwo;
public:
Base2( )
{
SampleDataTwo = 200;
}
~Base2( )
{
}
int SampleFunctTwo( )
{
return SampleDataTwo;
}
};
class Derived1 : public Base1, public Base2
{
int MyData;
public:
Derived1( )
{
MyData = 300;
}
~Derived1( )
{
}
int MyFunct( )
{
return (MyData + SampleDataOne + SampleDataTwo);
}
};
int main( )
{
Base1 SampleObjOne;
Base2 SampleObjTwo;
Derived1 SampleObjThree;
cout << SampleObjThree.Base1 :: SampleFunctOne( ) << endl;
cout << SampleObjThree.Base2 :: SampleFunctTwo( ) << endl;
return 0;
}
```

(A) 100

(B) 200

(C) Both A & B

(D) None of the mentioned

Q7. Which of the following can the derived class inherit?

Data members

Member functions

All of the listed options

None of the listed options

Q8. What will happen when introduce the interface of classes in a run-time polymorphic hierarchy?

Separation of interface from implementation

Merging of interface from implementation

Separation of interface from debugging

None of the mentioned

Q9. What is output of the following program?

```
1  class student
2  {
3     public : int marks;
4     void disp()
5     {
6        cout<<”its base class”
7     };
8     class topper:public student
9     {
10       public :
11       void disp()
12       {
13          cout<<”Its derived class”;
14       }
15    }
16    void main() { student s; topper t;
17    s.disp();
18    t.disp();
19 }
```

Its base classIts derived class

Its base class Its derived class

Its derived classIts base class

Its derived class Its base class

Q10. Which among the following best describes polymorphism?

It is the ability for a message/data to be processed in more than one form

It is the ability for a message/data to be processed in only 1 form

It is the ability for many messages/data to be processed in one way

It is the ability for undefined message/data to be processed in at least one way

Q11. Which type of function among the following shows polymorphism?

Inline function

Virtual function

Undefined functions

Class member functions

Q12. Find the output of the following program.

```
1  class education
2  {
3     char name[10];
4     public : disp()
5     {
6        cout<<”Its education system”;
7     }
8     class school:public education
9     {
10       public: void dsip()
11       {
12          cout<<”Its school education system”;
13       }
14    };
15    void main()
16    {
17             school s;
18             s.disp();
19    }
20 }
```

Its school education system

Its education system

Its school education systemIts education system

Its education systemIts school education system

Q13. If same message is passed to objects of several different classes and all of those can respond in a different way, what is this feature called?

- Inheritance
- Overloading
- Polymorphism
- Overriding

Q14. Which among the following is not true for polymorphism?

- It is feature of OOP
- Ease in readability of program
- Helps in redefining the same functionality
- Increases overhead of function definition always

Q15. If 2 classes derive one base class and redefine a function of base class, also overload some operators inside class body. Among these two things of function and operator overloading, where is polymorphism used?

- Function overloading only
- Operator overloading only
- Both of these are using polymorphism
- Either function overloading or operator overloading because polymorphism can be applied only once in a program

Q16. Which of the following correctly describes overloading of functions?

- Virtual polymorphism
- Transient polymorphism
- Ad-hoc polymorphism
- Pseudo polymorphism

Q17. Which among the following can show polymorphism?

- Overloading ||

- Overloading +=
- Overloading <<
- Overloading &&

Q18. What is the other name of compile-time polymorphism?

- Static polymorphism
- Dynamic polymorphism
- Executing polymorphism
- Non-executing polymorphism

Q19. Which among the following best defines single level inheritance?

- A class inheriting a derived class
- A class inheriting a base class
- A class inheriting a nested class
- A class which gets inherited by 2 classes

Q20. Which type of inheritance leads to diamond problem?

- Single level
- Multi-level
- Multiple
- Hierarchical

Answer Key & Solution

Section 1 - MCQ

Q1

All of the listed options

Solution

The benefit of dynamic binding and polymorphism is that they help making the code easier to extend but by multiple inheritance it makes harder to track.

Q2

Comma

Solution

For using multiple inheritance, simply specify each base class (just like in single inheritance), separated by a comma.

Q3

class name : access specifer class name

Solution

No Solution

Q4

40

Solution

In this program, We are multiplying the value 10 and 4 by using inheritance.

Output:

$ g++ des.cpp

$ a.out

40

Q5

(C) both A & B

Solution

As we can’t able to create the dynamic instance for the triangle, So it is arising an exception.

Output:

$ g++ exs3.cpp

$ a.out

Can’t do the dynamic_cast lor!!!

Caught: bad_cast exception. Myshape is not mytriangle.

Q6

(C) Both A & B

Solution

In this program, We are passing the values by using multiple inheritance and printing the derived values.

Output:

$ g++ mul4.cpp

$ a.out

100

200

Q7

All of the listed options

Solution

No Solution

Q8

Separation of interface from implementation

Solution

No Solution

Q9

Its base classIts derived class

Solution

No Solution

Q10

It is the ability for a message/data to be processed in more than one form

Solution

No Solution

Q11

Virtual function

Solution

No Solution

Q12

Its school education system

Solution

No Solution

Q13

Polymorphism

Solution

No Solution

Q14

Increases overhead of function definition always

Solution

No Solution

Q15

Either function overloading or operator overloading because polymorphism can be applied only once in a program

Solution

No Solution

Q16

Ad-hoc polymorphism

Solution

No Solution

Q17

Overloading <<

Solution

No Solution

Q18

Static polymorphism

Solution

No Solution

Q19

A class inheriting a base class

Solution

No Solution

Q20

Multiple

Solution

No Solution