Started on	Tuesday, 31 March 2020, 12:04 AM
State	Finished
Completed on	Tuesday, 31 March 2020, 12:27 AM
Time taken	23 mins 15 secs
Marks	9.67/12.00
Grade	**80.56** out of 100.00
Feedback	Congratulations!! You have passed by securing more than 80%

Question 1

Incorrect

Mark 0.00 out of 1.00

10. class FourWheeler {

11. protected ✖ void display() { }

12. }

13. class Car extends FourWheeler {

14. private ✖ void display() { }

15. }

Which method at line 14, will correctly complete class Car?

The visibility of a method can only be increased as we traverse down the hierarchy.

The correct answer is:

10. class FourWheeler {

11. [private] void display() { }

12. }

13. class Car extends FourWheeler {

14. [protected] void display() { }

15. }

Which method at line 14, will correctly complete class Car?

Question 2

Correct

Mark 1.00 out of 1.00

Given a method in a public class, private ✔ access modifier must be used to restrict access to that method to only the other members of the same class.

Your answer is correct.

The correct answer is:
Given a method in a public class, [private] access modifier must be used to restrict access to that method to only the other members of the same class.

Correct

Mark 1.00 out of 1.00

A default member of a super class in one package cannot ✔ be accessed in its own subclass which is in a different package.

A protected member of a super class in one package can ✔ be accessed by its own subclass which is in a different package.

Your answer is correct.

The correct answer is:

Constructor of the superclass [can] be invoked by from its subclass.

A default member of a super class in one package [cannot] be accessed in its own subclass which is in a different package.

A protected member of a super class in one package [can] be accessed by its own subclass which is in a different package.

Correct

Mark 1.00 out of 1.00

```
class FourWheeler
{
   public FourWheeler()
   {
      System.out.println("Class FourWheeler");
   }
}
class Car extends FourWheeler
{
   public Car()
   {
      System.out.println("Class Car");
   }
}
class Audi extends Car
{
  public Audi()
   {
      super();
      System.out.println("Class Audi");
   }
}
class Driver
{
   public static void main(String args[])
   {
      Audi cc=new Audi();
   }
}
```

Select one:

- a. Exception occurs
- b. Class Audi
Class Car
Class FourWheeler
- c. Compile Time Error
- d. Class FourWheeler
Class Car
Class Audi ✔

The first statement that always gets executed from within any constructor is super() which means the invocation of super class no-parameterized constructor.

FourWheeler is the parent of Car and Car is the parent of Audi. The no-parameterized constructor call of Audi happens from the driver class. Followed by the no-parameterized constructor call of Car. Followed by the no-parameterized constructor call of FourWheeler. Hence, the output "Class Four WheelerClass Car Class Audi".

The correct answer is:
Class FourWheeler
Class Car
Class Audi

Correct

Mark 1.00 out of 1.00

Select one:

a. Computer, AppleComputer and IBMComputer are sibling classes.

b. Computer is a superclass, AppleComputer is a subclasses of Computer, and IBMComputer is a sublclas of AppleComputer

c. Computer is the super class, AppleComputer and IBMComputer are subclasses of computer ✔

d. IBMComputer is the superclass, AppleComputer and Computer are subclasses of IBMComputer.

Your answer is correct.

The correct answer is: Computer is the super class, AppleComputer and IBMComputer are subclasses of computer

Question 6

Correct

Mark 1.00 out of 1.00

Given:

```
class FourWheeler
{
public FourWheeler ()
{
System.out.print(1);
}
}
class Car extends FourWheeler
{
public Car()
{
System.out.print(2);
}
}
class Audi extends Car
{
public Audi()
{
System.out.print(3);
}
}
public class Driver
{
public static void main( String[] argv )
{
new Audi();
}
}
```

What is the result when this code is executed?

Select one:

a. The code runs with no output

b. 321

c. 3

d. 123 ✔

The first statement that always gets executed from within any constructor is super() which means the invocation of super class no-parameterized constructor.

FourWheeler is the parent of Car and Car is the parent of Audi. The no-parameterized constructor call of Audi happens from the driver class. Followed by the no-parameterized constructor call of Car. Followed by the no-parameterized constructor call of FourWheeler. Hence, the output "123".

The correct answer is: 123

Question 7

Partially correct

Mark 0.67 out of 1.00

Interpret which of the following statements are correct with respect to inheritance relationship in java?

Select one or more:

- [x] a. object of subclass referenced by super class type can invoke overridden sub class methods ✔
- [ ] b. object of subclass referenced by super class type can access super class variables
- [ ] c. object of subclass referenced by super class type can access newly defined sub class variables
- [ ] d. object of subclass referenced by super class type can invoke newly defined sub class methods
- [x] e. object of subclass referenced by super class type can invoke super class methods ✔

Your answer is partially correct.

You have correctly selected 2.
The correct answers are: object of subclass referenced by super class type can invoke super class methods, object of subclass referenced by super class type can invoke overridden sub class methods, object of subclass referenced by super class type can access super class variables

Question 8

Correct

Mark 1.00 out of 1.00

Default Access ✔ is the most restrictive access modifier that will allow members of one class to have access to members of another class in the same package.

Public | Synchronized | Protected | Abstract

Your answer is correct.

The correct answer is:

[Default Access] is the most restrictive access modifier that will allow members of one class to have access to members of another class in the same package.

Incorrect

Mark 0.00 out of 1.00

```
{
  //logic with return statement
}
}
class Car extends FourWheeler
{
    protected void getObject() {}
}
class Driver
{
public static void main(String args[])
{
   FourWheeler object = new Car();
   object.getObject();
}
}
```

✖

private FourWheeler getObject() {} | protected Car getObject() {} | Car getObject() {}

Java 5.0 onwards it is possible to have different return type for a overriding method in child class, but child's return type should be sub-type of parent's return type. The visibility of a method can only be increased as we traverse down the hierarchy.

The correct answer is: class FourWheeler{

```
protected FourWheeler getObject()
{
  //logic with return statement
}
}
class Car extends FourWheeler
{
   [protected Car getObject() {} ]
}
class Driver
{
public static void main(String args[])
{
   FourWheeler object = new Car();
   object.getObject();
}
}
```

Correct

Mark 1.00 out of 1.00

TemporaryEmployee are declared in another package **subordpack**. The basicPay attribute should be accessed only by means of a derived class object.

How to ensure that the basicPay attribute is not accessed directly by the other classes in the subordpack?

Employee.java

package mainpack ✔ **;**

public class Employee{

protected ✔ **int basicPay;**

}

/*PermanentEmployee.java*/

package subordpack ✔ **;**

public class PermanentEmployee extends Employee{

.....

}

//TemporaryEmployee.java

package subordpack ✔ **;**

public class TemporaryEmployee extends Employee{

.....

}

For a child class that is residing in a package to access a variable of its parent that is residing in a different package, the variable in the parent has to be declared "protected" so that it will be visible to all its children across packages. The "protected" basicPay attribute is not accessed directly by the other classes in the subordpack, in our case.

The correct answer is:

The class Employee is declared in a package **mainpack** and the Derived classes PermanentEmployee and TemporaryEmployee are declared in another package **subordpack**. The basicPay attribute should be accessed only by means of a derived class object.

How to ensure that the basicPay attribute is not accessed directly by the other classes in the subordpack?

Employee.java
package [mainpack];
public class Employee{
[protected] int basicPay;
}
/*PermanentEmployee.java*/
package [subordpack];
public class PermanentEmployee extends Employee{
.....
}
//TemporaryEmployee.java
package [subordpack];
public class TemporaryEmployee extends Employee{
.....
}

Question 11

Correct

Mark 1.00 out of 1.00

State True or False
Child class objects can be instantiated when the parent class constructor is protected

Select one:

- (•) True ✔
- ( ) False

The correct answer is 'True'.

Correct

Mark 1.00 out of 1.00

```
 private void add(int operand1, int operand2)
 {
   System.out.println(operand1 + operand2);
 }
}
public class Addition extends ArithmeticOperation
{
 public void show()
 {
   add(10, 12);
 }
  public static void main(String args[])
 {
   Addition ob = new Addition();
    ob.show();
 }
}
```

What will be the output of above code when compiled and executed?

Select one:

- a. Runtime error as add method is not defined in MethodOverriding class
- b. Will compile and display 32
- c. Compile time error ✔
- d. Will compile and display 1012
- e. Will print false

The method add cannot be invoked from anywhere since it's declared private

The correct answer is: Compile time error