Day: M T W T F S

Date: __/__/__

IQRA National University
Peshawar

Name — Amir Sohail

ID — 16436

Department — BS(CS)

Semester — 2th

Subject — OOPs

PaPer — Final

Submitted to M. Ayub Khan

Data — 29/6/2020

Day: M T W T F S

Date: / 1 /

Question 1 (a)

Access Modifier Used in java:

Access Modifier are like entry gates for other class. A class can control what information or data can be accessible by other classes.

java Provide a number of access modifiers to help you set the level of access you want for class as well as the field, method and construction in your classes. A member has Package or default accessibility when no accessibility modifier is ~~acces~~ specified.

There two type of modifiers.

access modifier and non-access modifier.

Access modifier are many type.

Private access modifiers

Default access modifiers

Protected access modifiers

Public access modifiers

Explain in detail Private and Default access modifiers.

M T W T F S

Date: / 2 /

Private access Modifiers ⇒

The access level of a Private modifier is only within the class. It cannot be accessed form outside the class.

The scope of Private modifiers

i Private Data member and methods are only accessible within the class.

ii class and interface cannot be declared as Private.

iii If a class has Private constructor then you cannot create the object the that class from outside of the class.

Example of Private access modifiers.

This example throws compilation error because we are trying to access the Private data member and method of class ABC in the class Example. The Private data member and method are only accessible within the class.

```
class ABC
{
    Private double num = 100;
    Private int square (int a)
{
    return a * a ;
    }
    }
    Public class
    {
Public static Void main (String args[])
{
    ABC obj = new ABC;
System.out.Println (obj.num);
System.out.Println (obj.square (10));
    }
```

outPut

ComPile = time error

Day: M T W T F S

Date: / /04

Default access modifiers =>

The access level of a default modifiers is only within the Packege. It cannot be access from outside the Package. if you do not specify any access level, it will be default.

when we do not mention any access modifier, it is called default access modifiers. The scope of this modifier is limited to the Package only. This mean that if we have a class with the default access modifier in a Package, only those classes that are in this Package can access his class. No other class outside this Package can access this class. similarly, if we have a default method or data member in a class, it would not be visible in the class of another Package.

Example of Default access modifiers:

Day: M T W T F S

Date: / 5 /

```
Package abc Package;
Public class Addition
{
int add Two Number (inta, intb)
{
   return a+b;
}
}
Package xyz Package;
import abc Package.*;
Public class Test
{
Public Static Void main (String arg[]);
   Addition obj = new Addition ();
   obj. add Two Number (10, 21);
}
}
```

out Put

Error

add Two number (int, int) from the type is not visible at xyzPackage .Test. main (Test. java : 12)

Diary: M T W T F S

Date: 8/6

Question 1 (b)

Program

Default and Private

```
Package P1;
class Geek
{
    void display()
    {
System.out.Println("Hellow world!");
    }
}
Packag 2;
class GeekNew
{
Public static void main(String arg[])
{
Greek obj = new Greek();
    object display();
    }
    }
```

output = time error

Package p1;

class A

{

private void display ()

{

System.out.println("Greek for Greek");

}

}

class B

{

Public static void main(String arg[])

{

A obj = new A();

obj.display();

}

}

output

error: display() has Private access in object display.

Private and default access modifier Program..

Day: M T W T F S

Date: / 8 / ___

Question 2 (a)

Public and Protect access Modifiers.

Public access modifier ⇒

The access level of a Public modifier is everywhere. It can be accessed from within the class, outside the class, within the Package and outside the Package. The members, methods and classes that are declare Public can be accessed from anywhere. This modifiers does not put any restriction on the class.

Example of Public access modifiers

Lets take the same example that we have seen above but this time the method add Two Number () has Public modifiers and class Test is able to access this method without even extending the Addition class. This because Public modifier has visibility everywhere.

Day: M T W T F S

Date: __/9/__

```
Package abc Package;
Public class Addition
{
Public int addTwo Number(int a, int b)
    return a+b;
}
}
```

Test java

```
Package xyzPackage;
imPort abcPackage.*;
class Test
{
Public static void main (String args[]){
    Addition obj = new addition();
System.out.Println(obj.addTwo Number(100,1));
}
```

outPut

Compile= 101

Day: ☐ M ☐ T ☐ W ☐ T ☐ F ☐ S

Date: / 10

Protected Access Modifier ⇒

The access level of a Protect modifier is within the Package and outside Package through child class. If you do not make the child class. It cannot be access from outside the Package.

Protected data member and method are only accessible by the class of the same Package and the subclass Present in any Package. You can also say that the Protected access modifier with one exception that it has visibility is sub classes.

classes cannot be declared Protected. This access modifier is generally used in a Parent child relationship

Example of Protected access Modifiers

In this example the class Test which is Present in another Package is able to call the addTwoNumber() method, which is declared protected.

Day M T W T F S

Date: / 11 / __

This is because the Test class extend class Addition and the Protect modifiers allows the access of Protect members in subclass.

```
Package abcPackage;
Public class Addition
{
Protected int addTwoNumber(int a, int b)
{
    return a+b;
}
```

Test.java

```
Package xyzPackage;
import abcPackage.*;
class Test extend Addition
{
Public static void (String args[])
    Test obj = new Test();
System.out.Println(obj.addTwoNumber(11, 22));
}
}
```

outPut = 33

Day: ☐M ☐T ☐W ☐T ☐F ☐S

Date: 12/6/__

Question 2 (b)

Protected and Public Access modifier Programs

```
Package P1;
Public class A
  {
Protected void display ()
  {
System.out.Println("Greek for Greek");
  }
}

Package P2;
class B extend A
{
public static void main(string arg [])
  {
    B obj = new B ();
    obj.display ();
  }
```

output

Greek for Greek

Day: M T W T F S Date: 13

```
Package P1;
Public class A
  {
  Public void display()
    {
System.out.Print("Greek for Greeks");
    }
  }
  Package P2;
  import P1.*;
    class B
{
Public static void main(String arg[])
  {
    A obj = new A;
    obj.display();
  }
}
```

OutPut = Greek for Greek

Protected and Public Program.

Diary M T W T F S

Date: /14/

Question 3 (a)

Inheritance =>

Inheritance can be defined as the Process where one class acquires the Properties of another. With the use of inheritance the information is made manageable in a hierarchical order. The class which inherits the Properties of other is know as subclass (derived class, child class) and the class whose Properties are inherited is know as superclass.

extend keyword =>

extends is the keyword used to inherit the Properties of a class.

Syntax;

```
class super {
    ....
    ....
}
class sub extend super {
    ....
    ....
}
```

Day: M T W T F S

Date: / 15

Use inheritance ⇒

on of the most important concepts in object oriented Programming is that of inheritance. Inheritance allows us to define a class in terms of another class which make it easier to create and maintaine an application. This also Provide an opportunity to reuse the code functionality, and fast implementation time. when creating a class, instead of writing completely new data member and member function, the Programming can designate that the new class should inherit the member of an existing class. This existing class is called the base class and the new class is referred to as the derived class. The idea of inheritance implements the is a relationship.

Day: M T W T F S

Date: / 16 /

Java Inheritance Example:

Example of inheritance Doctor and Surgeon.

```
class Doctor
{
void Doctor detail ()
{
System.out.Println("Doctor * Detail..");
}
}
class Surgeon extend Doctor
{
Void Surgeon Detail()
{
System.out.Println("Surgeon Detail...");
}
}
Public class Hospital
{
Public static void main(String arg[])
{
Surgeon S = new Surgeon();
```

Date: /17/

M T W T F S

```
S. Doctor - Detail ();
S. Surgeon - Detail ();
}
}
```

Advantage of inheritance in OOPs

But one may argue that across all classes, you have a repeated pieces of code The function are not required to be implemented individually. This is inheritance java.

M T W T F S

Date: /18/

Question 3 (b)

Inheritance class Animal

```
class Animal
{
void eat () { System.out.Println ("eating.");
 }
 }
class Dog extends Animal
 {
void bark () {System.out.Println("barking..");
 }
 }
 class Cat extends Animal
  {
void meow () {System.out.Println("meowing..");}}
 }
class Test Inheritance 3
 {
Public static void main(String arg[])
 {
 Cat c = new Cat ();
```

Day: M T W T F S

Date: / 19

```
c.meow();
c.eat ();
  }
}
```

output

Meowing...

eating..

when two or more class inherits single class, it is know hierarchical inheritance. Dog and Cat classes inherits the Animal class. So there is hierarchical inheritance.

Day: M T W T F S

Date: 20/__

Question 4 (a)

Polymorphism ⇒

Polymorphism is the ability of an object to take on many form. The most common use of Polymorphism in OOPs occure when a Parent class reference is used to refer to a child class object. Any Java object that can Pass more than one IS-A test is considered to be Polymorphism. In Java, all Java object are Polymorphic since any object will Pass IS.A test for their own type and for the class object. It is important to know that the only Possible way to access an object is through a reference variable. A reference variable can be only one type. once declared, the type of a reference variable cannot be change.

The reference variable can be

Day: M T W T F S

Date: 21/

reassigned to other object provided that it is not declared final.

The type of the reference variable would determine the method that it can invoke on the object.

A reference variable can refer to any object of its declared type. A reference variable can be declared as a class or interface type.

Example:

Public interface vegetarian { }

Public class Animal { }

Publi class Deer extend Animal

Now, the Deer class is considered to be Polymorphic since this has multiple inheritance. following are true example.

- A Deer IS-A Animal
- A Deer IS-A vegetarian
- A Deer IS-A Deer
- A Deer IS-A object.

Diary: M T W T F S

Date: / 22

when we apply the reference variable facts to a Deer object reference, the following, declaration are legal.

Example

Deer d = new Deer ();

Animal a = d;

Vegetarian v = d;

object o = d;

All the referrence variable d, a v, o refer to the Same Deer object in the heap.

Day: M T W T F S

Date: 23/

Question 4 (b)

Program

```
/* file name : Employee.java */
    Public class Employee
    {
  Private String name;
  Private String address;
  Private int number;
Public Employee(String name, String
      address, int Number)
      {
System.out.Println("Constructing an Employee");
    this.name = name;
    this.address = address;
    this.number = number;
    }
  Public void mailcheck()
    {
System.out.Println("Mailing a check to" +
  this.name + " " + this address);
  }
  Public String tostring()
```

Day: M T W T F S

Date: 24

```
{
    return name + "  " + address
    + "  "  + number;
    }
    Public  string get Name()
    {
      return  name ;
      }
  Public  String get Address ()
    {
      return  address ;
    }
  Public void set Address (String newAddress)
  {
      address = newAddress;
    }
  Public int getNumber()
    {
    return number ;
    }
    }
```

Now suppose we extend Employees class

Date: 25

M T W T F S

```
/* File name : Salary.java */
Public class Salary extend EmPlyee
{
  Private double Salary;
  Public Salary (String name, String
address, int number, double Salary)
  {
    SUPer (name, address, number);
      setSalary (Salary);
  }
  Public void ~~chec~~ mailcheck()
  {
System.out.Println("within mailcheck of
    salary class");
System.out.Println("mailing check
  to " + getName() + "with
Salary " + Salary);
  }
  Public double getSalary()
  {
    return Salary;
  }
```

Day: M T W T F S Date: 26/

```
Public void getSalary (double newSalary)
  {
    if (newSalary >= 0.0)
    {
      Salary = newSalary;
    }
  }
  Public double computePay ()
    {
System.out.Println("Computing Salary Pay
for" + getName() );
    return Salary / 52;
    }
  }
```

Now, you study the following Program carefully and try to determine its output

```
/* File Name : VirtualDemo.java */
  Public class virtualDemo
    {
Public static void main (String[] args)
    {
```

Day: M T W T F S

Date: / 27

```
Salary s = new Salary("Mohd Mohtashim"
, "Ambehta UP", 3, 3600.00);
Employee e = new Salary("John
Adam", "Boston", MA", 2,
2400.00);
System.out.Println("Call mailcheck
using Salary reference");
S.mailcheck();
System.out.Println("\n mailCheck
using Employee reference>>);
e.mail Check();
}
}
```

output

Constructing an Employee
Constructing an Employee
Call mailcheck using Salary reference
within mailcheck of Salary class
Mailing Check to Mohd Mohtashim with
Salary 3600.00

Day: ☐ M ☐ T ☐ W ☐ T ☐ F ☐ S

Date: 28

call mailcheck Using Employee reference within mailcheck of Salary class mailing check to John Adam with Salary 2400.0

Here, we instantiate two Salary object. one using a Salary reference s. and the other using Employee reference.

Date: /29

Day: M T W T F S

Question 5 (a)

Abstraction ⇒

Abstraction is the quality of dealing with idea rather than events. For example, when you consider the case of e-mail complex detail such such as what as what happens as soon as you send e mail, the protocol your e-mail server user are hidden from the user. Therefore, to send an e-mail you just need to type the content, mention the address of the receiver, and click send. Likewise in OOPs abstraction is a process of hiding the implemention detail from the users, only the funictionally will be provided to the user. In other words, the user will have the information on

Day: M T W T F S

Date: 30

what the object does instead of how it does it.

Abstract class

A class which contain the abstract keyword in its declaration is know as abstract.

- Abstract class may or may not contain abstract method, i.e method without body (Public void get();
- But if a class has at least one abstract methods then the class must be declar abstract.
- If a class is declare abstract, it cannot be instantiated.
- To use an abstract class you have to inherit it from another class, provide, implementation to the abstract method in it.
- If you inherit an abstract class you have to provide implemention to all the abstrat meth in it

Day: M T W T F S

Date: /31/

Question = 5 (b)

Abstract Program

```
abstract class Animal
{
Public abstract void animal sound();
  Public void sleeP ()
    {
System. out. Println ("Zzz");
    }
    }
class Pig extends Animal
    {
Public void animal sund ()
    {
System. out. Println ("The Pig say : wee wee");
    }
    }
class My main class
    {
Public static Void main (String [] arg)
    {
```

Date: 32

Day M T W T F S

```
Pig myPig = new Pig ();
myPig. animal Sound ();
myPig. sleep ();
}
}
```

outPut

The Pig says : wee wee

ZZZ

END