NAME: IBNE AMIN

ID NO: 15830

PROGRAM: BS,SE2

[redacted]

Department of Computer Science

Final Term Exam Spring 2020

Subject: Object Oriented Programming

BS (CS,SE) Instructor: M.Ayub Khan

There are total 5 questions in this paper. Max Marks:50

Each question carry equal marks.
Please answer briefly.

Q1. a. Why access modifiers are used in java, explain in detail Private and Default access modifiers?

Java provides a number of access modifiers to help you set the level of access you want for classes as well as the fields, methods and constructors in your classes. A member has package or default accessibility when no accessibility modifier is specified.

Access Modifiers in Java

There are two types of modifiers in Java: access modifiers and non-access modifiers.

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package.

1) Private

The private access modifier is accessible only within the class.

Simple example of private access modifier

In this example, we have created two classes A and Simple. A class contains private data member and private method. We are accessing these private members from outside the class, so there is a compile-time error.

Role of Private Constructor

If you make any class constructor private, you cannot create the instance of that class from outside the class. For example:

---

2) Default

If you don't use any modifier, it is treated as by default. The default modifier is accessible only within package. It cannot be accessed from outside the package. It provides more accessibility than private. But, it is more restrictive than protected, and public.

Example of default access modifier

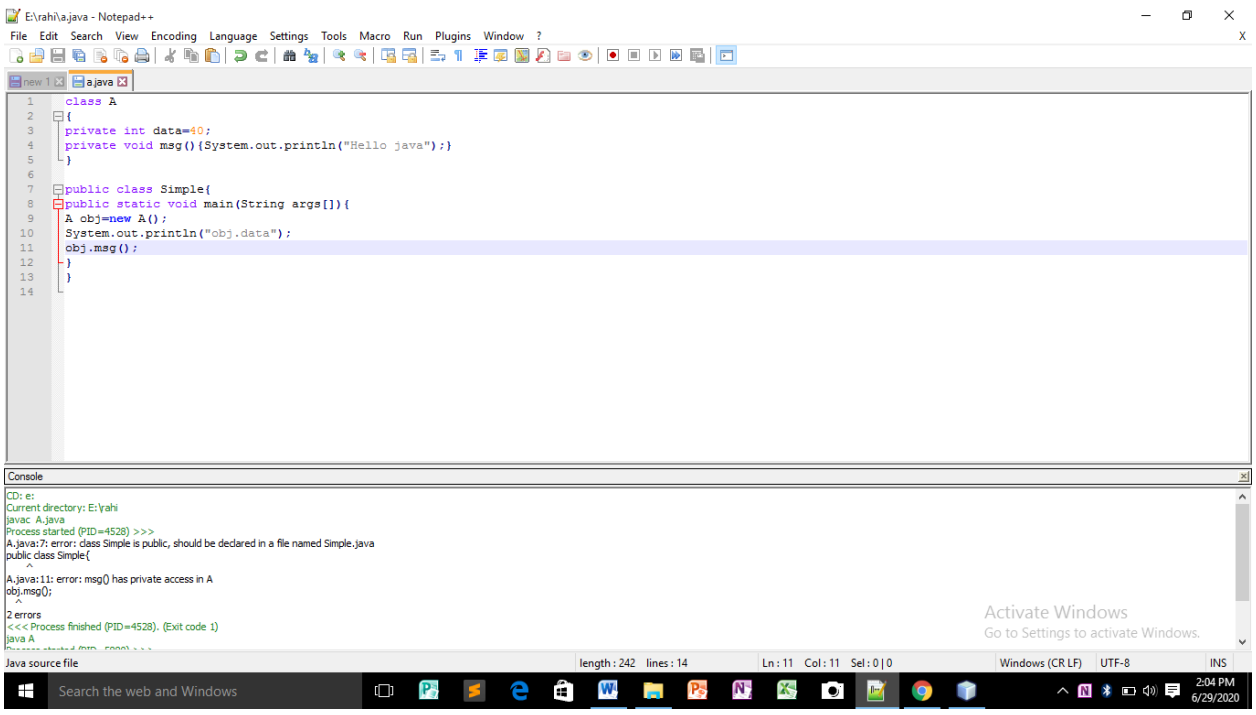
In this example, we have created two packages pack and mypack. We are accessing the A class from outside its package, since A class is not public, so it cannot be accessed from outside the package.

In the above example, the scope of class A and its method msg() is default so it cannot be accessed from outside the package.

b. Write a specific program of the above mentioned access modifiers in java.

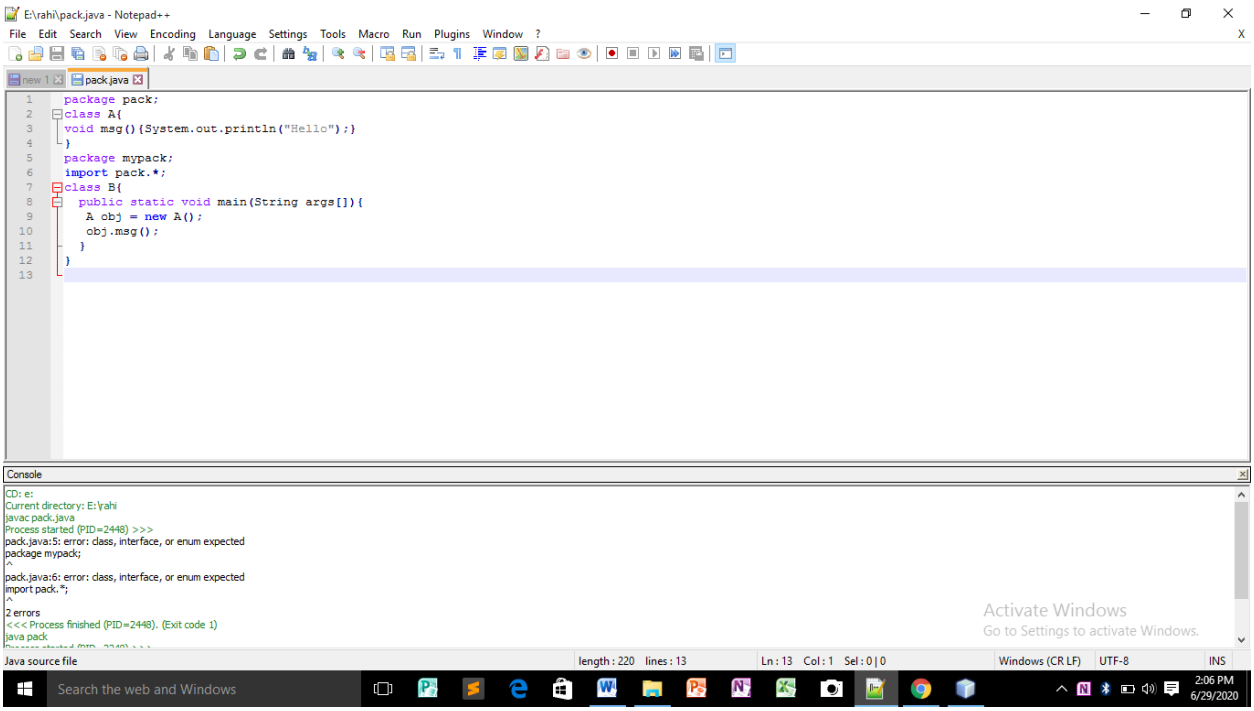
1) Private

```
class A{
private int data=40;
private void msg(){System.out.println("Hello java");}
}

public class Simple{
public static void main(String args[]){
A obj=new A();
System.out.println(obj.data);
obj.msg();
}
}
```



2) Default

```
package pack;
class A{
void msg(){System.out.println("Hello");}
}
package mypack;
import pack.*;
class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```



Q2. a. Explain in detail Public and Protected access modifiers?

4. Public access modifier

The members, methods and classes that are declared public can be accessed from anywhere. This modifier doesn't put any restriction on the access.

public access modifie in java

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

3. Protected Access Modifier

Protected data member and method are only accessible by the classes of the same package and the subclasses present in any package. You can also say that the protected access modifier is similar to default access modifier with one exception that it has visibility in sub classes.
Classes cannot be declared protected. This access modifier is generally used in a parent child relationship.

Protected access modifier example in Java

In this example the class Test which is present in another package is able to call the addTwoNumber(method, which is declared protected. This is because the Test class extends class Addition and the protected modifier allows the access of protected members in subclasses (in any packages).

b. Write a specific program of the above mentioned access modifiers in java.

Public access modifier

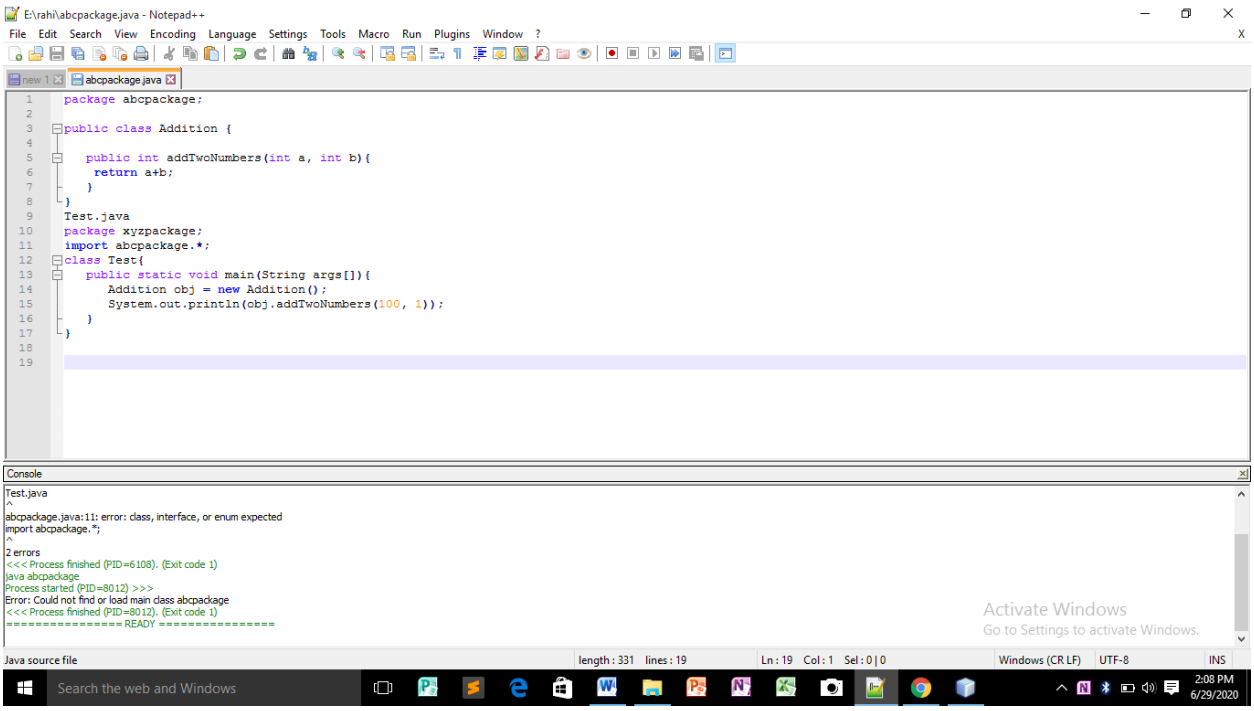
```
package abcpackage;
```

```
public class Addition {


   public int addTwoNumbers(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.addTwoNumbers(100, 1));
   }
```

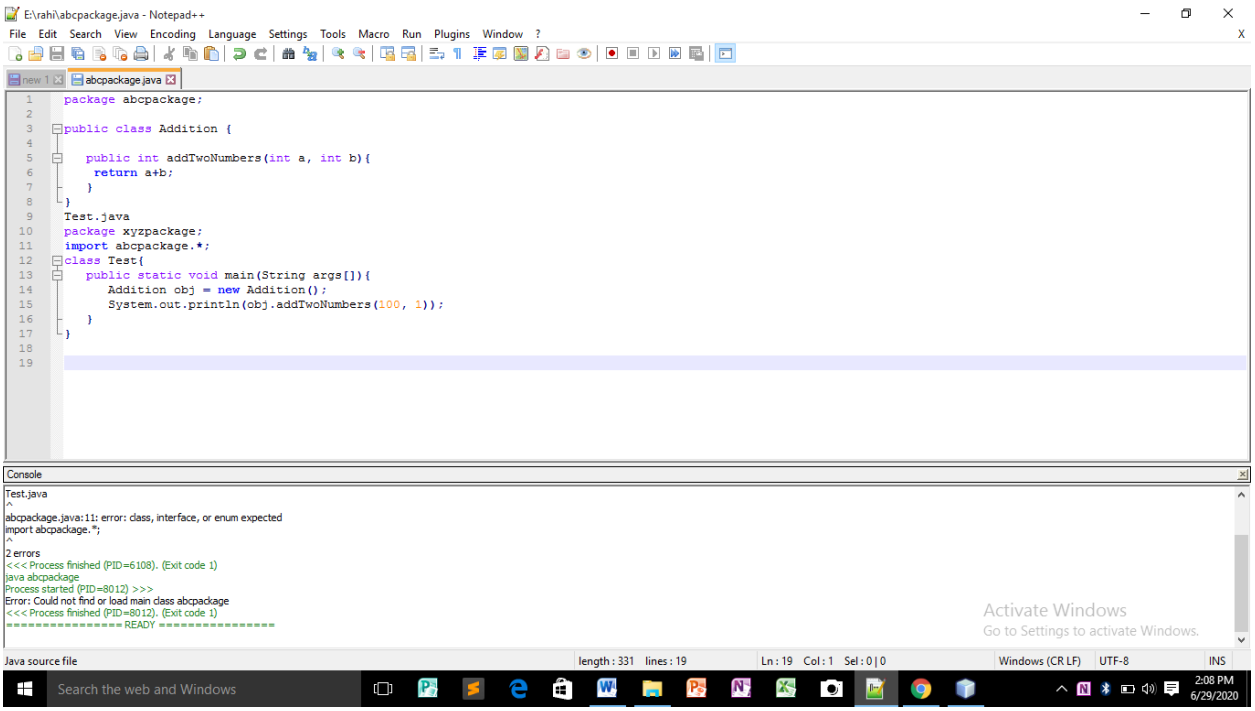
}



Protected access modifier example in Java

```
package abcpackage;
public class Addition {

  protected int addTwoNumbers(int a, int b){
        return a+b;
  }
}
package xyzpackage;
```

```java
import abcpackage.*;
class Test extends Addition{
  public static void main(String args[]){
      Test obj = new Test();
      System.out.println(obj.addTwoNumbers(11, 22));
  }
}
```



Q3. a. What is inheritance and why it is used, discuss in detail ?

In object-oriented programming, **inheritance** is the mechanism of basing an object or class upon another object (prototype-based inheritance)or

class (class-based inheritance), retaining similar implementation. Also defined as deriving new classes (sub classes) from existing ones such as super class or base class and then forming them into a hierarchy of classes. In most class-based object-oriented languages, an object created through inheritance, a "child object", acquires all the properties and behaviors of the "parent object" , with the exception of: constructors, destructor, overloaded operators and friend functions of the base class. Inheritance allows programmers to create classes that are built upon existing classes, to specify a new implementation while maintaining the same behaviors (realizing an interface), to reuse code and to independently extend original software via public classes and interfaces. The relationships of objects or classes through inheritance give rise to a directed graph.

Types

There are various types of inheritance, based on paradigm and specific language.[7]

Single inheritance

where subclasses inherit the features of one superclass. A class acquires the properties of another class.

Multiple inheritance

where one class can have more than one superclass and inherit features from all parent classes.

Multilevel inheritance

where a subclass is inherited from another subclass. It is not uncommon that a class is derived from another derived class as shown in the figure "Multilevel inheritance"

Hierarchical inheritance

This is where one class serves as a superclass (base class) for more than one sub class. For example, a parent class, A, can have two subclasses B and C. Both B and C's parent class is A, but B and C are two separate subclasses.

Hybrid inheritance

Hybrid inheritance is when a mix of two or more of the above types of inheritance occurs. An example of this is when class A has a subclass B which has two subclasses,

b. Write a program using Inheritance class on Animal in java.

```
class Employee{
 float salary=40000;
}
class Programmer extends Employee{
 int bonus=10000;
 public static void main(String args[]){
   Programmer p=new Programmer();
   System.out.println("Programmer salary is:"+p.salary);
   System.out.println("Bonus of Programmer is:"+p.bonus);
}
}
```

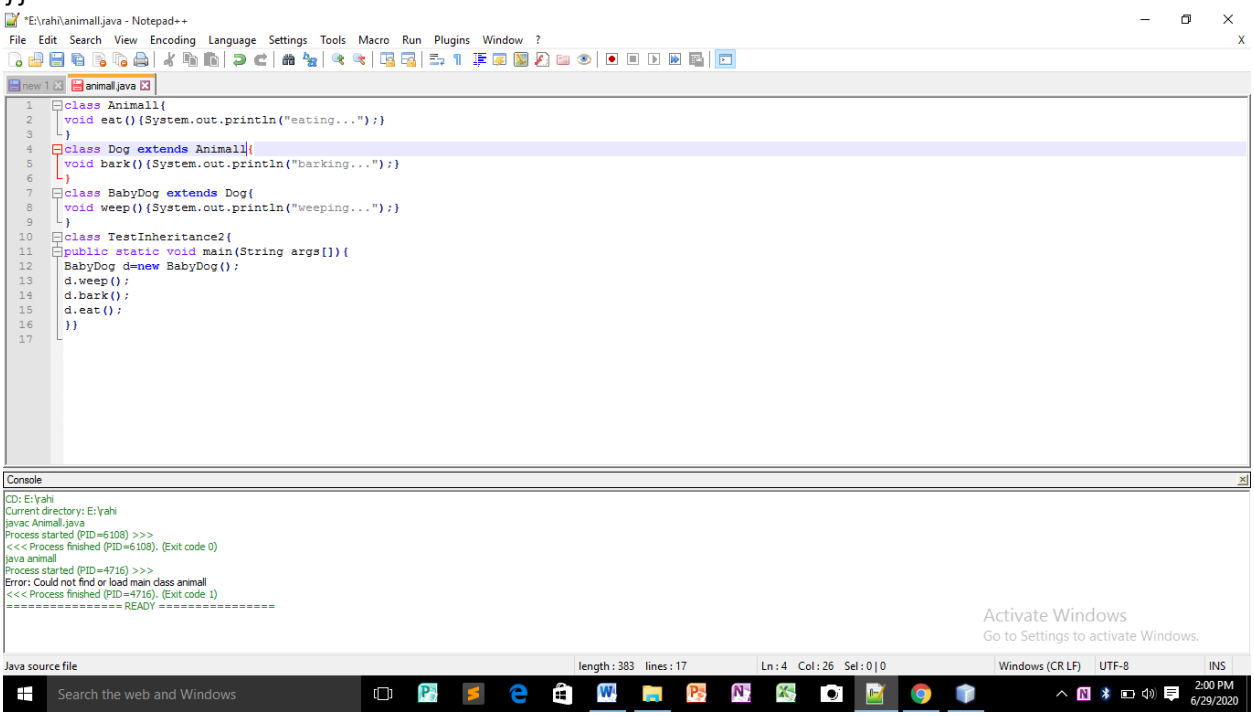
Single Inheritance Example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
```

Multilevel Inheritance Example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
```

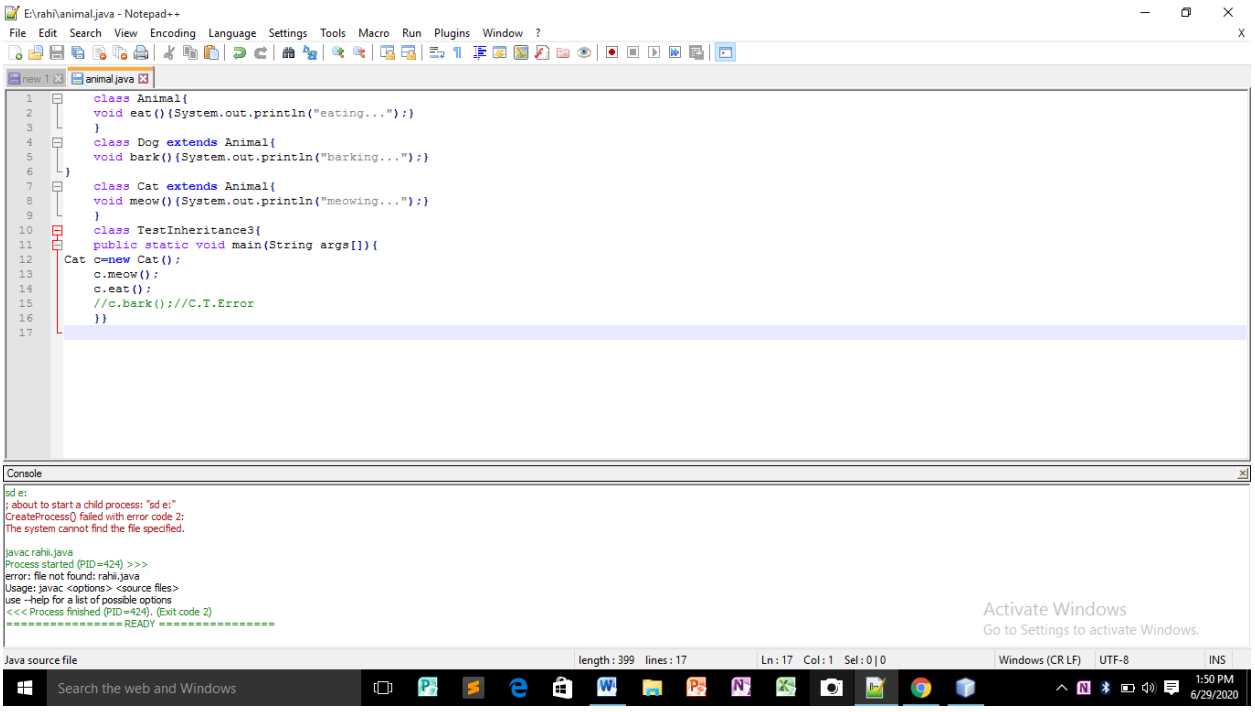
```
d.bark();
d.eat();
}}
```



Hierarchical inheritance

```
1. class Animal{
2. void eat(){System.out.println("eating...");}
3. }
4. class Dog extends Animal{
5. void bark(){System.out.println("barking...");}
6. }
7. class Cat extends Animal{
8. void meow(){System.out.println("meowing...");}
9. }
10.        class TestInheritance3{
```

```
11.     public static void main(String args[]){
12.     Cat c=new Cat();
13.     c.meow();
14.     c.eat();
15.     }}
```



Q4. a. What is polymorphism and why it is used, discuss in detail ?

polymorphism

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs 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 polymorphic. In Java, all Java objects are polymorphic since any object will pass the 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 of only one type. Once declared, the type of a reference variable cannot be changed.

The reference variable can be reassigned to other objects provided that it is not declared final. The type of the reference variable would determine the methods that it can invoke on the object.

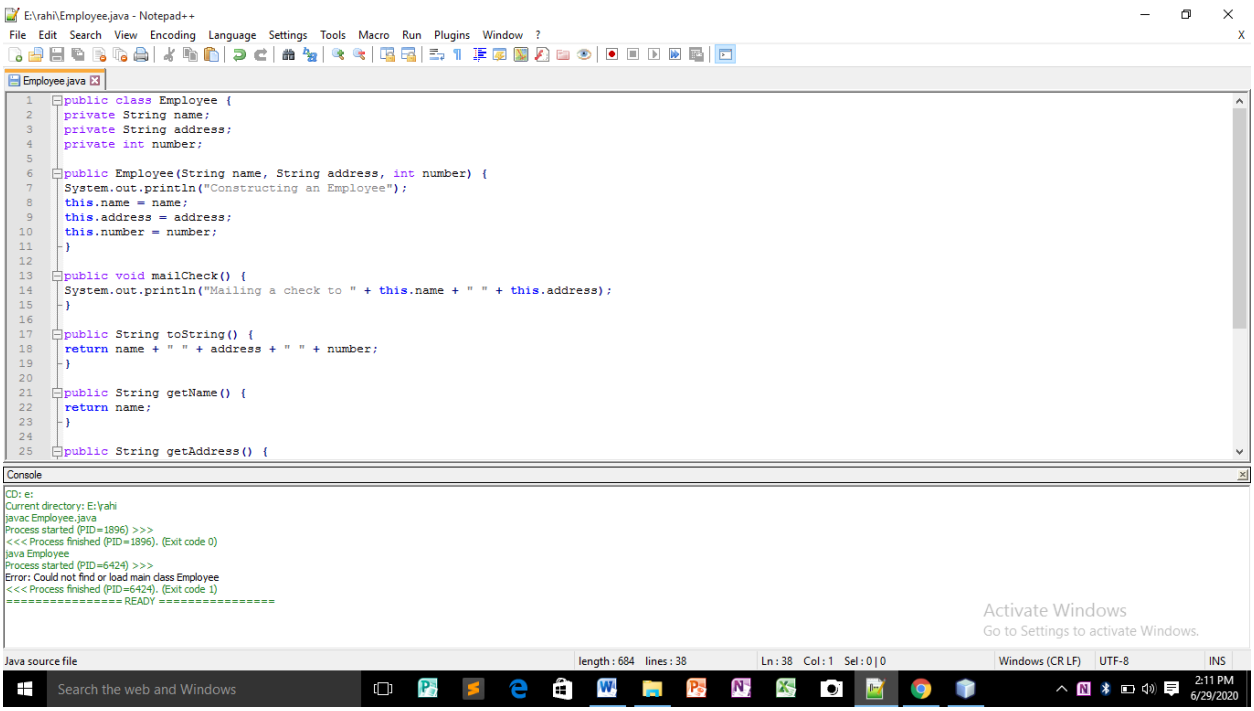
A reference variable can refer to any object of its declared type or any subtype of its declared type. A reference variable can be declared as a class or interface type.

b. Write a program using polymorphism in a class on Employee in java.

```
/* 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() {
        return name + " " + address + " " + number;
}

public String getName() {
        return name;
}
```

```
public String getAddress() {
        return address;
}

public void setAddress(String newAddress) {
        address = newAddress;
}

public int getNumber() {
        return number;
}
}
```



Q5. a. Why abstraction is used in OOP, discuss in detail ?

What is Abstraction in OOP?

Abstraction is selecting data from a larger pool to show only the relevant details of the object to the user. Abstraction "shows" only the essential attributes and "hides" unnecessary information. It helps to reduce programming complexity and effort. It is one of the most important concepts of OOPs.

What is Abstraction in Java?

Abstraction in JAVA "shows" only the essential attributes and "hides" unnecessary details of the object from the user. In Java, abstraction is accomplished using Abstract classes, Abstract methods, and interfaces. Abstraction helps in reducing programming complexity and effort.

Abstract Class

A class which is declared "abstract" is called as an abstract class. It can have abstract methods as well as concrete methods. A normal class cannot have abstract methods.

Abstract Method

A method without a body is known as an Abstract Method. It must be declared in an abstract class. The abstract method will never be final because the abstract class must implement all the abstract methods.

Rules of Abstract Method

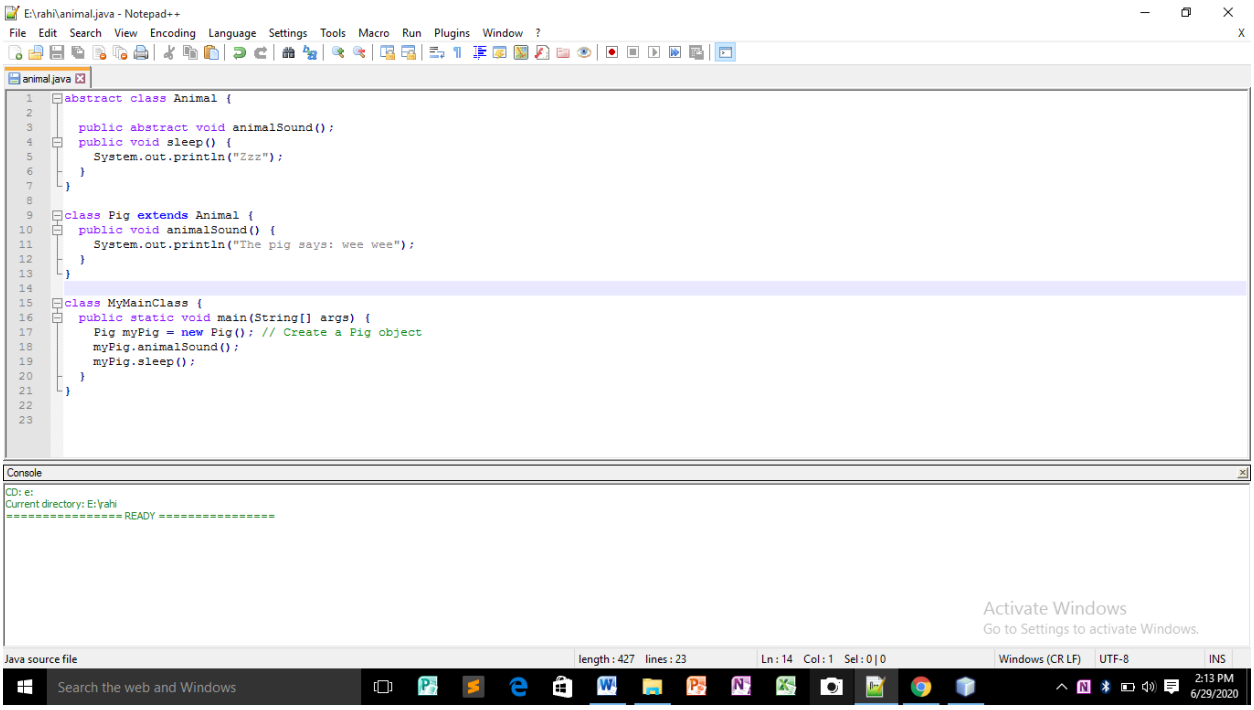
- Abstract methods do not have an implementation; it only has method signature

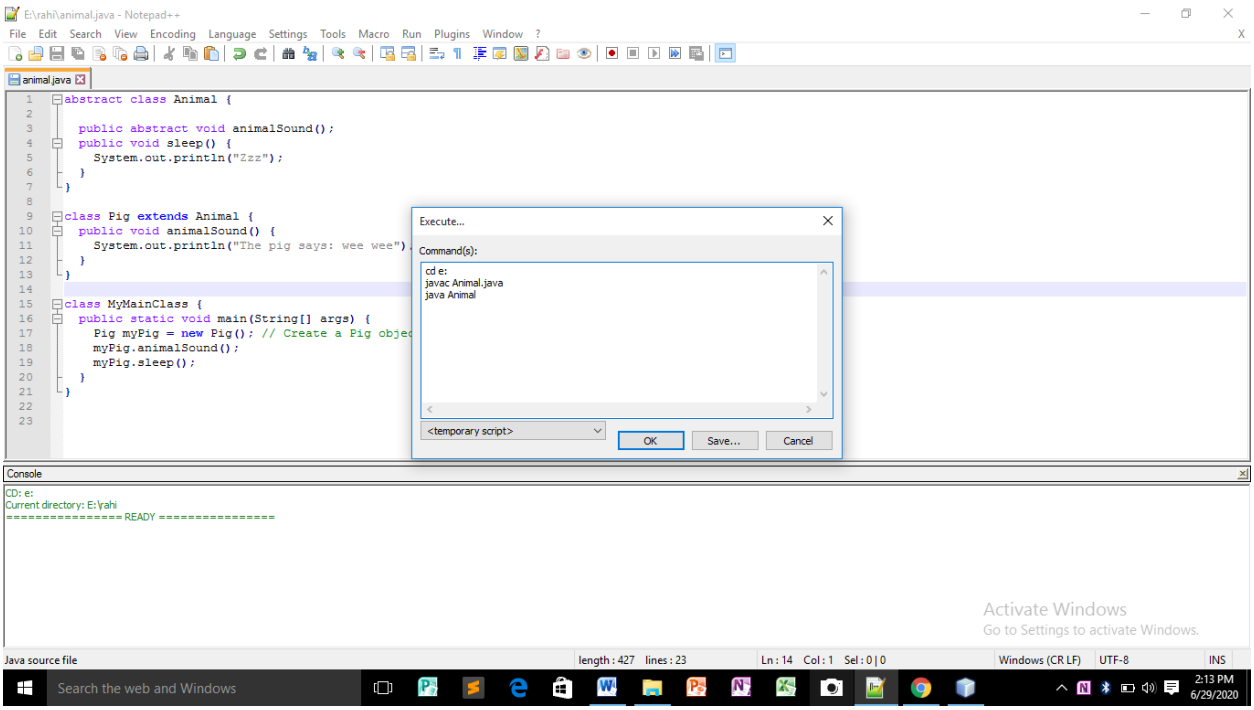
- If a class is using an abstract method they must be declared abstract. The opposite cannot be true. This means that an abstract class does not necessarily have an abstract method.
- If a regular class extends an abstract class, then that class must implement all the abstract methods of the abstract parent

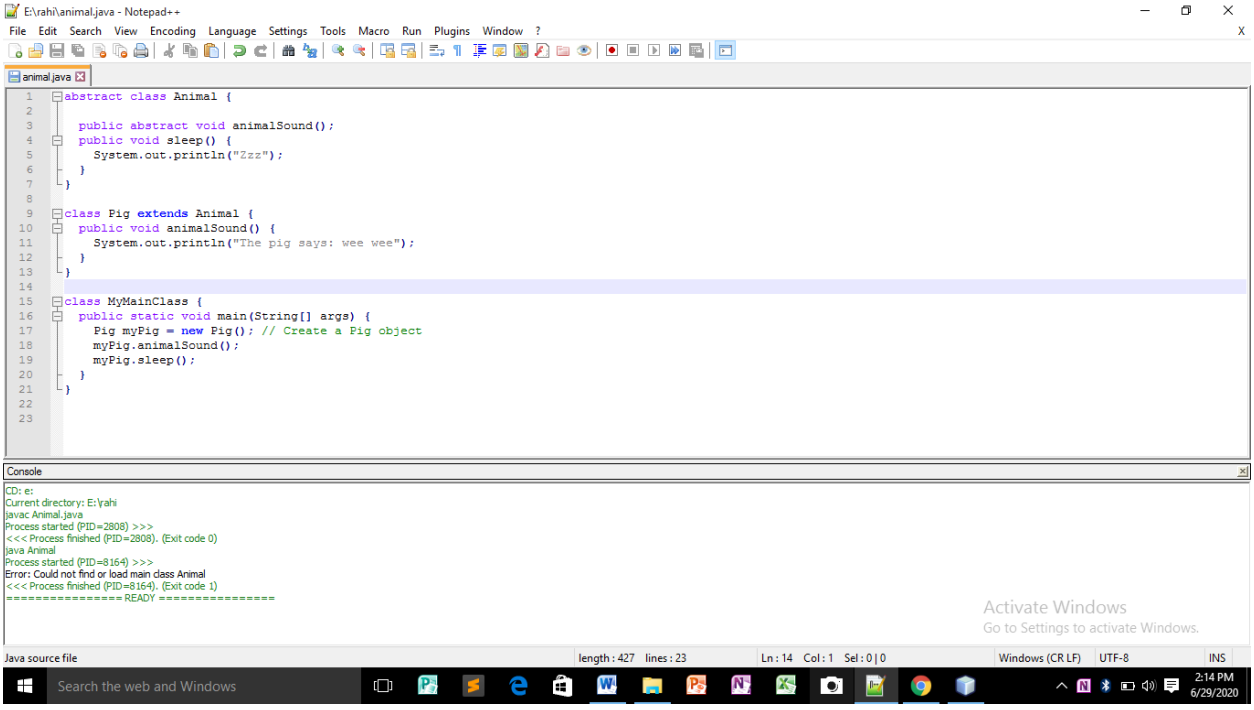
b. Write a program on abstraction in java.

```
abstract class Animal {

  public abstract void animalSound();
  public void sleep() {
    System.out.println("Zzz");
  }
}

class Pig extends Animal {
  public void animalSound() {
    System.out.println("The pig says: wee wee");
  }
```

```
}


class MyMainClass {
  public static void main(String[] args) {
    Pig myPig = new Pig(); // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```




E:\rahi\animal.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
animal.java
abstract class Animal {
public abstract void animalSound();
public void sleep() {
System.out.println("Zzz");
}
}
class Pig extends Animal {
public void animalSound() {
System.out.println("The pig says: wee wee");
}
}
class MyMainClass {
public static void main(String[] args) {
Pig myPig = new Pig(); // Create a Pig obje
myPig.animalSound();
myPig.sleep();
}
}
Execute...
Command(s):
cd e:
javac Animal.java
java Animal
<temporary script>
OK
Save...
Cancel
Console
CD: e:
Current directory: E:\rahi
READY
Activate Windows
Go to Settings to activate Windows.
Java source file
length : 427 lines : 23
Ln : 14 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Search the web and Windows
2:13 PM
6/29/2020


E:\rahi\animal.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
animal.java
abstract class Animal {
public abstract void animalSound();
public void sleep() {
System.out.println("Zzz");
}
}
class Pig extends Animal {
public void animalSound() {
System.out.println("The pig says: wee wee");
}
}
class MyMainClass {
public static void main(String[] args) {
Pig myPig = new Pig(); // Create a Pig object
myPig.animalSound();
myPig.sleep();
}
}
Console
CD: e:
Current directory: E:\rahi
javac Animal.java
Process started (PID=2808) >>>
<<< Process finished (PID=2808). (Exit code 0)
java Animal
Process started (PID=8164) >>>
Error: Could not find or load main class Animal
<<< Process finished (PID=8164). (Exit code 1)
READY
Activate Windows
Go to Settings to activate Windows.
Java source file
length : 427 lines : 23
Ln : 14 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Search the web and Windows
2:14 PM
6/29/2020