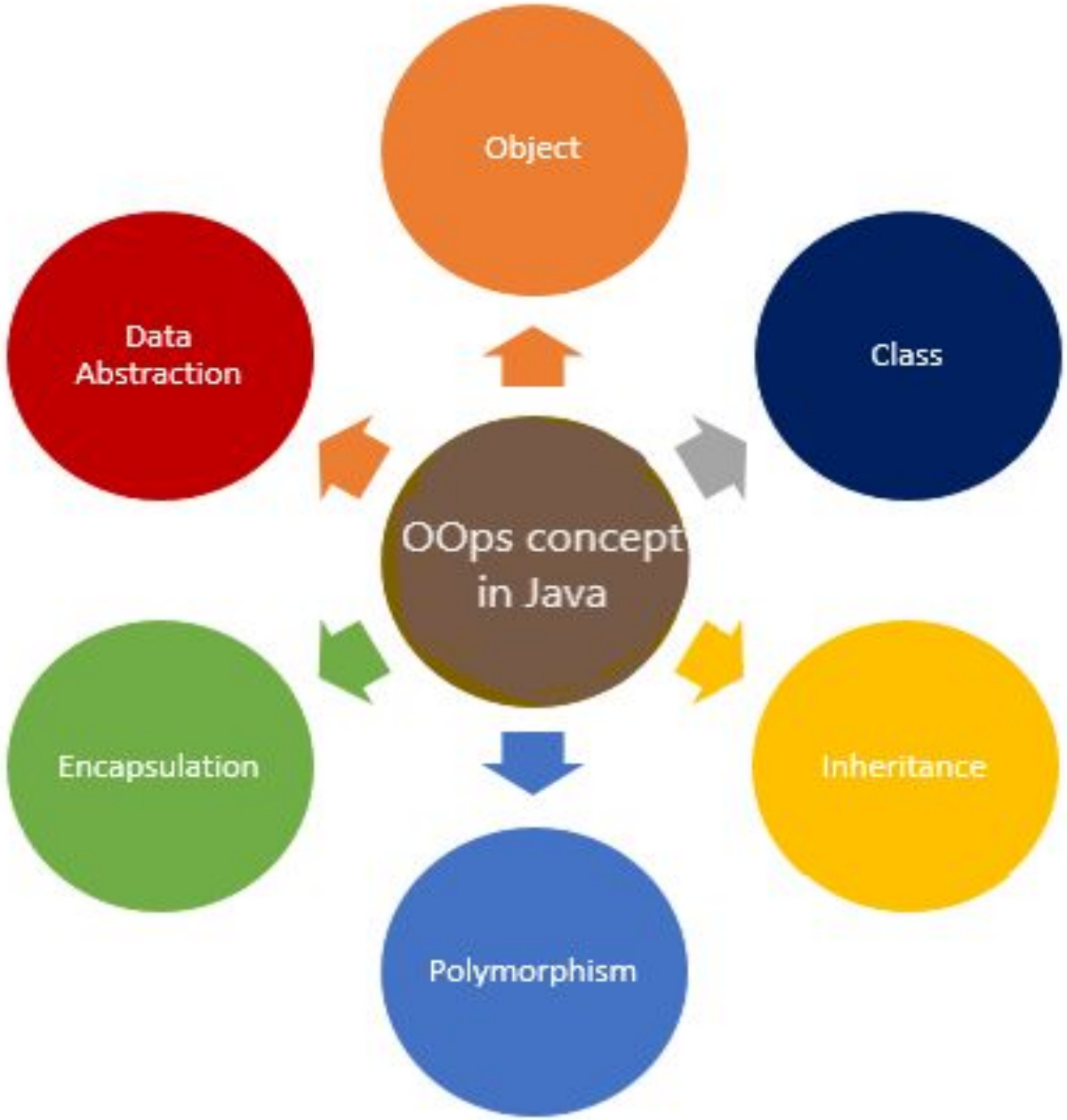
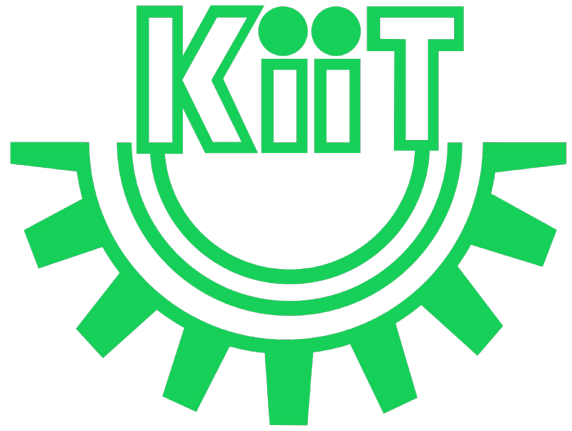


CS20004: Object Oriented Programming using Java

Lec-15

In this Discussion . . .

- Access control Mechanism
- Import Statement
- Name conflict
- Short Vs. Full References
- Static Import
- References



Access control mechanism

- Access control is a mechanism, an attribute of encapsulation which restricts the access of certain members of a class to specific parts of a program.
- Access to members of a class can be controlled using the **access modifiers**.
- The access modifiers in Java specify 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 suitable access modifier on it.

Access control mechanism

- There are four access modifiers in Java:
  - **Private:** The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
  - **Default:** The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
  - **Protected:** The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
  - **Public:** 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.

Access control mechanism

- There are many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc.

Access Modifier	Within Class	Within Package	Outside Package by Subclass only	Outside Package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

What is the use of Access Modifiers in Java?

- As specified earlier, access modifiers are used to restrict the access of members of a class.
- Let's understand the need with the help of an example. Let's consider an employee class as shown below:

```
class Employee
{
        int  empid;
        String  empname;
        float  salary;
        //Methods which operate on above data
members
}
```

What is the use of Access Modifiers in Java?

- By looking at the code provided below we can say that the access modifier for all the three data members is default.

```
class Employee
{
	int  empid;
	String  empname;
	float  salary;
	//Methods which operate on above data
members
}
```

What is the use of Access Modifiers in Java?

- As members with default (also known as Package Private or no modifier) access modifier are accessible throughout the package (in other classes), a programmer might, by mistake, try to make an employee's salary negative as shown below:

```
Employee e1 = new Employee();
e1.salary = -1000.00;
```

What is the use of Access Modifiers in Java?

- **Although above code is syntactically correct, it is logically incorrect.**
- To prevent such things to happen, in general, all the data members are declared private and are accessible only through public methods.
- So, we can modify our Employee class as shown below:

```
class Employee
{
	private int empid;
	private String emp;
	private float salary;
	public void setSalary(float sal)
	{
		if(sal < 0)
		{
	System.out.println("Salary cannot be negative");
		}
		else
		{
			salary = sal;
		}}
	//Other methods
}
```

What is the use of Access Modifiers in Java?

- By looking at the below code, we can say that one can access the salary field only through *setSalary()* method. Now, we can set the salary of an employee as shown below:

```
Employee e1 = new Employee();
e1.setSalary(-1000.00);  //Gives error as salary cannot be negative
e1.setSalary(2000.00);  //salary of e1 will be assigned 2000.00
```

The modified Employee class is the best way of writing programs in Java.

Private Access Modifier

- The private access modifier is accessible only within the class.
- In the example below, we have created two classes Accexp and Privacc. Accexp class contains private data member and private method. We are accessing these private members from outside the class.

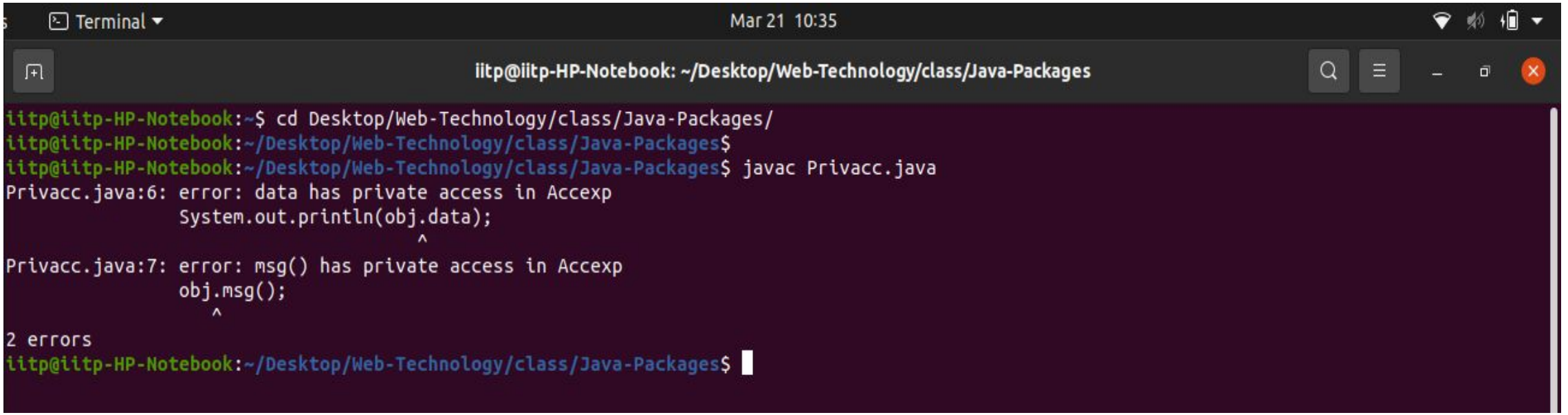
Note: A class cannot be private or protected except nested class.

```
class Accexp
{
        private int data=40;
        private void msg()
        {
                System.out.println("Showcasing the usage
scenarios of the private access modifier");
        }
}
```

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

Private Access Modifier

- The private access modifier is accessible only within the class.
- In the example below, we have created two classes Accexp and Privacc. Accexp 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 we make any class constructor private, then we cannot create the instance of that class from outside the class. For example:

```
class Privcons
{
	//private constructor
	private Privcons()
	{

	}
	void msg()
	{
		System.out.println("Demo Program for showing the
role of private constructor");
	}
}
```

```
public class Privconsexp
{
	public static void main(String args[])
	{
		Privcons obj=new Privcons();
	}
}
```

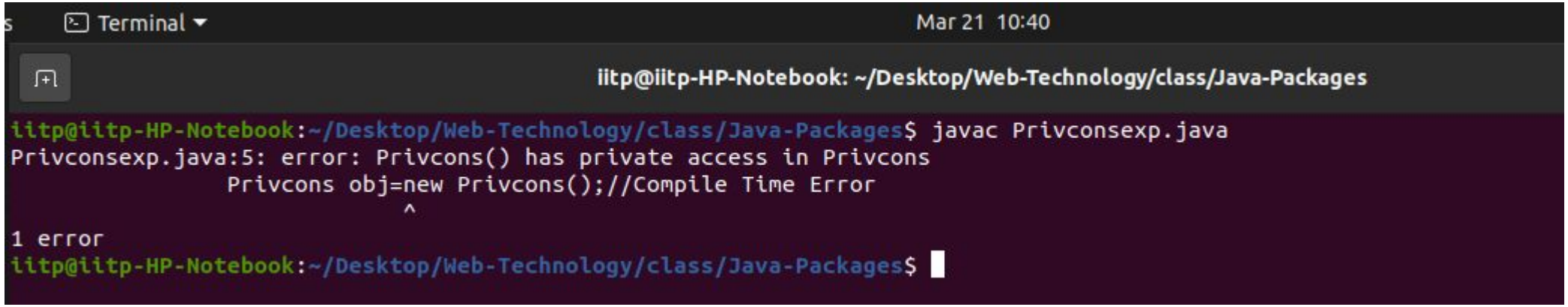
Role of Private Constructor

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

```
class Privcons
{
        //private constructor
        private Privcons()
        {

        }
        void msg()
        {
                System.out.println("Demo Program for showing the
role of private constructor");
        }
}
```

```
public class Privconsexp
{
        public static void main(String args[])
        {
                Privcons obj=new Privcons();//Compile Time Error
        }
}
```



Default Access Modifier

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

```
package defpack;
class Defacc
{
        void msg()
        {
                System.out.println("Explaining the default access
modifier");
        }
}
```

```
package defaultpack;
import defpack.*;
class Defaultaccexp
{
        public static void main(String args[])
        {
                Defacc obj = new Defacc();
                obj.msg();
        }
}
```

Default Access Modifier

- In the below program(s), we have created two packages defpack and defaultpack. We are accessing the Defacc class from outside its package

```
package defpack;
class Defacc
{
        void msg()
        {
                System.out.println("Explaining the default access
modifier");
        }
}
```

```
package defaultpack;
import defpack.*;
class Defaultaccexp
{
        public static void main(String args[])
        {
                Defacc obj = new Defacc();
                obj.msg();
        }
}
```

Default Access Modifier

- In the below program(s), we have created two packages defpack and defaultpack.
- We are accessing the Defacc class from outside its package. As Defacc class is not public, so it cannot be accessed from outside the package.

```
package defpack;
class Defacc
{
        void msg()
        {
                System.out.println("Explaining the default access
modifier");
        }
}
```

```
package defaultpack;
import defpack.*;
class Defaultaccexp
{
        public static void main(String args[])
        {
                Defacc obj = new Defacc();
                obj.msg();
        }
}
```

Default Access Modifier

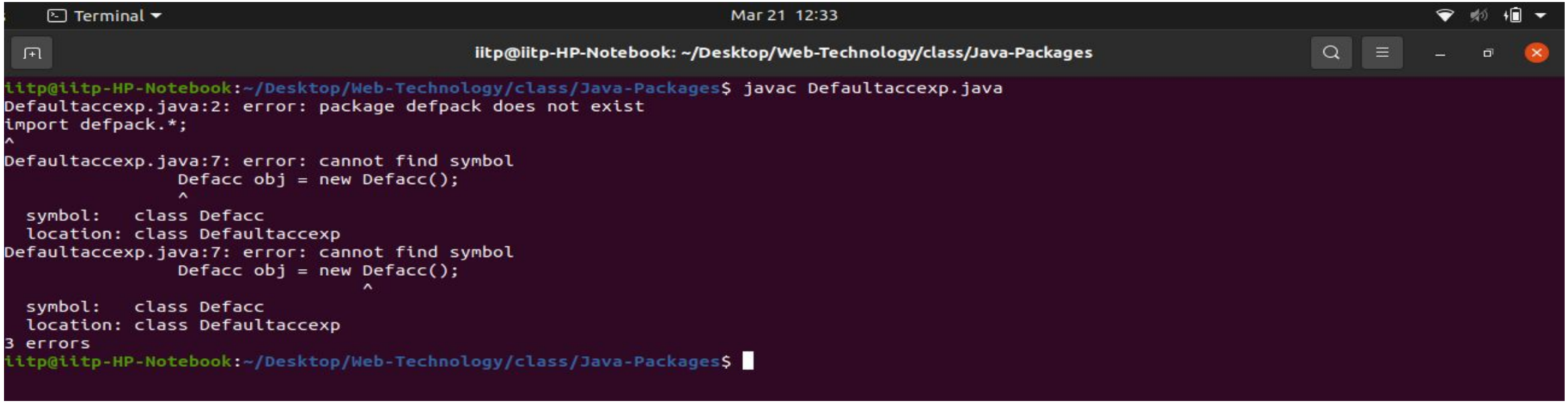
- In the below program(s), we have created two packages defpack and defaultpack.
- We are accessing the Defacc class from outside its package. As Defacc class is not public, so it cannot be accessed from outside the package.
- The scope of class Defacc and its method msg() is default so it cannot be accessed from outside the package.
- Thus, trying to compile the Defaultaccexp.java file results in a compile time error.

```
package defpack;
class Defacc
{
        void msg()
        {
                System.out.println("Explaining the default access
modifier");
        }
}
```

```
package defaultpack;
import defpack.*;
class Defaultaccexp
{
        public static void main(String args[])
        {
                Defacc obj = new Defacc();
                obj.msg();
        }
}
```

Default Access Modifier

- In the below program(s), we have created two packages defpack and defaultpack.
- We are accessing the Defacc class from outside its package. As Defacc class is not public, so it cannot be accessed from outside the package.
- The scope of class Defacc and its method msg() is default so it cannot be accessed from outside the package. Thus, trying to compile the Defaultaccexp.java file results in a compile time error.



Protected Access Modifier

- The **protected** access modifier is accessible within package and outside the package but through inheritance only.
- The **protected** access modifier can be applied on the data member, method and constructor. It can't be applied on the class.
- *We cannot declare a top-level class as private or protected. It can be either public or default (no modifier)*
- The **protected** access modifier provides more accessibility than the default modifier.

Protected Access Modifier

- In the below example, we have created the two packages protpack and protectedpack.
- The Protacc class of protpack package is public, so it can be accessed from outside the package.
- But the msg() of this package is declared as protected, so it can be accessed from outside the class only through inheritance.

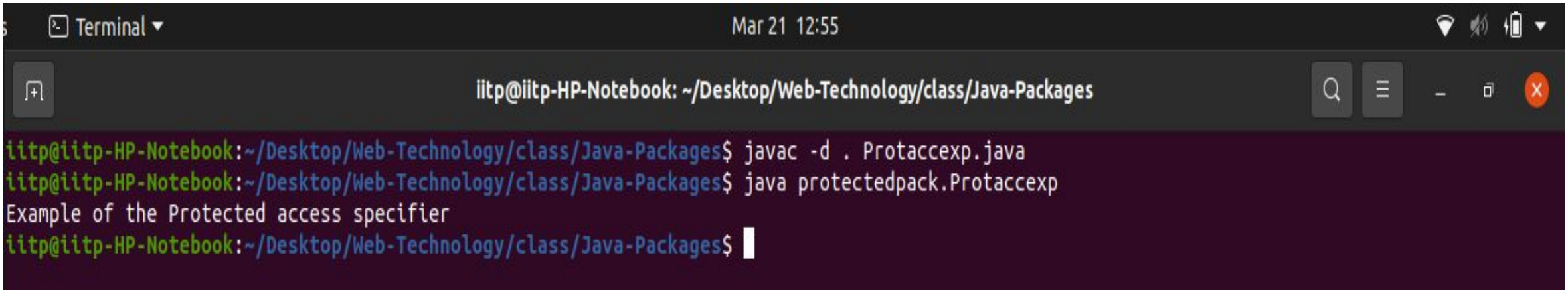
```
package protpack;
public class Protacc
{
        protected void msg()
        {
                System.out.println("Example of the Protected
access specifier");
        }
}
```

```
package protectedpack;
import protpack.*;

class Protaccexp extends Protacc
{
        public static void main(String args[])
        {
                Protaccexp obj = new Protaccexp();
                obj.msg();
        }
}
```

Protected Access Modifier

- The Protacc class of protpack package is public, so it can be accessed from outside the package.
- But the msg() of this package is declared as protected, so it can be accessed from outside the class only through inheritance.



Public Access Modifier

- The public access modifier is accessible everywhere. It has the widest scope among all other modifiers.

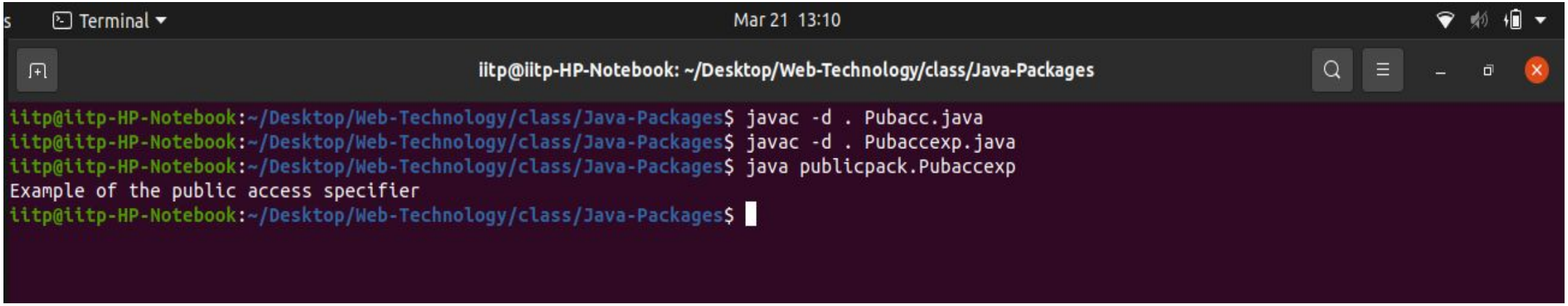
```
package pubpack;
public class Pubacc
{
        public void msg()
        {
                System.out.println("Example of the public access
specifier");
        }
}
```

```
package publicpack;
import pubpack.*;

class Pubaccexp
{
        public static void main(String args[])
        {
                Pubacc obj = new Pubacc();
                obj.msg();
        }
}
```

Public Access Modifier

- The public access modifier is accessible everywhere. It has the widest scope among all other modifiers.



Java Access Modifiers with Method overriding

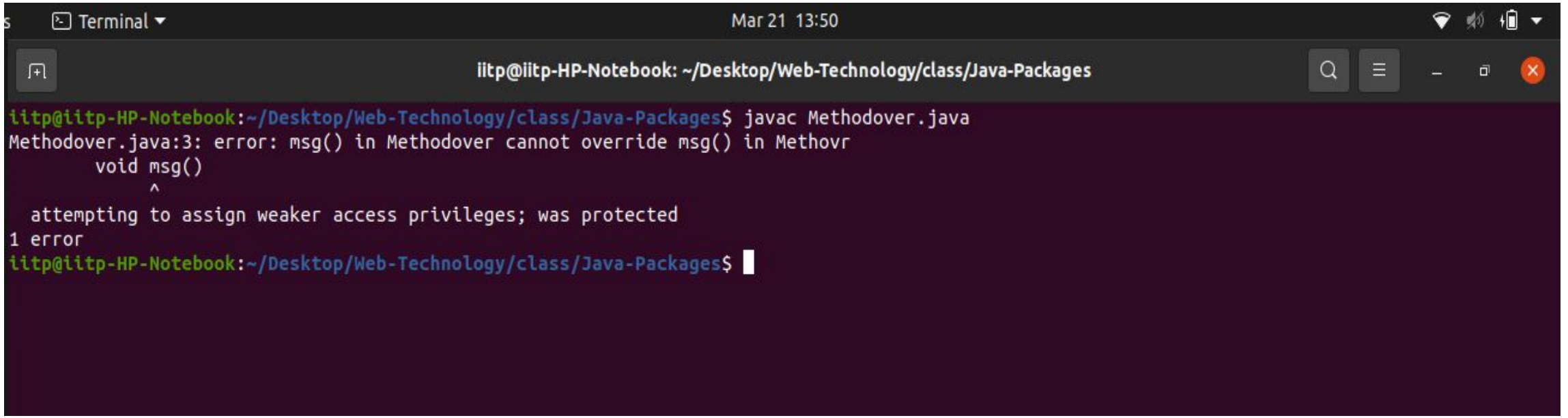
- If we are overriding any method, then overridden method (i.e. declared in subclass / child class) must not be more restrictive.

```
class Methovr
{
        protected void msg()
        {
                System.out.println("Example of method
overriding with concern to access specifiers");
        }
}
```

```
public class Methodover extends Methovr
{
        void msg()
        {
                System.out.println("Overriding");
        }
        public static void main(String args[])
        {
                Methodover obj=new Methodover();
                obj.msg();
        }
}
```

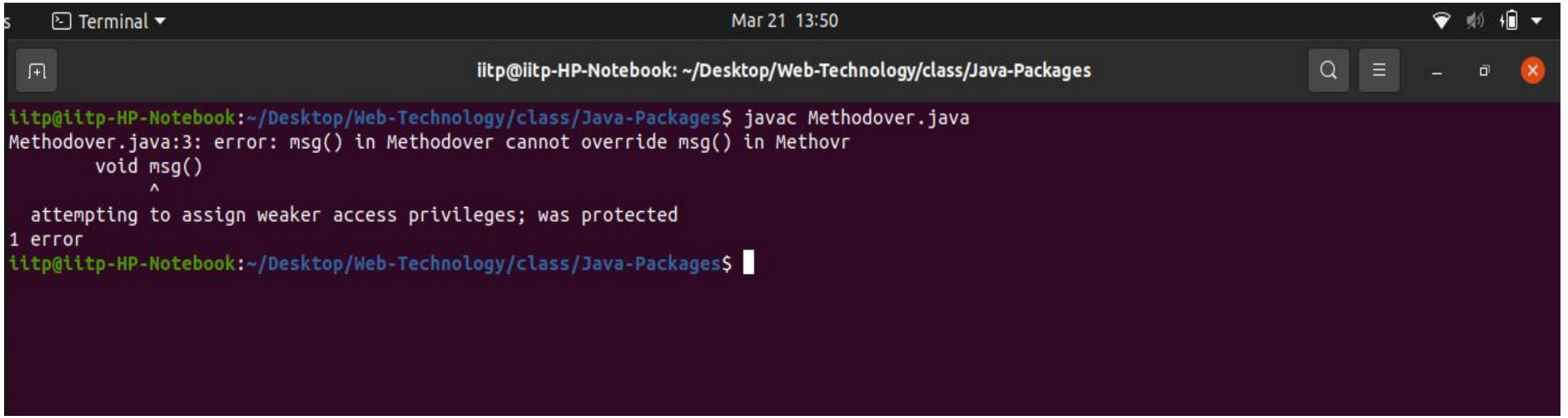
Java Access Modifiers with Method overriding

- If we are overriding any method, then overridden method (i.e. declared in subclass / child class) must not be more restrictive.



Java Access Modifiers with Method overriding

- If we are overriding any method, then overridden method (i.e. declared in subclass / child class) must not be more restrictive.



The default modifier is more restrictive than protected. That is why, there is a compile-time error.

Import statement

- The import statement occurs immediately after the package statement and before the class statement:

  package myPackage;
  import otherPackage1.otherPackage2.otherClass;
  class myClass{…}

- The Java system accepts this import statement by default:
  import java.lang.;*

- This package includes the basic language functions. Without such functions, Java is of no much use

Name Conflict

```
package otherPackage1;
class otherClass{ ... }
```

```
package otherPackage2;
class otherClass{ ... }
```

```
import otherPackage1.*;
import otherPackage2.*;
class myClass{
        ...
        otherClass
        ...
        }
```

Name Conflict

- Compiler will remain silent, unless we try to use otherClass. Then it will display an error message
- In this situation we should use the full name.

```
import otherPackage1.*;
import otherPackage2.*;
class myClass{
        ...
        otherPackage1.otherClass
        ...
        otherPackage2.otherClass
        ...
}
```

Short Vs. Full Reference

- Short reference:

  import java.util.;*

  class MyClass extends Date {…}
- Full Reference:

  class MyClass extends java.util.Date {…}

- *Only the public components in imported package are accessible for non-sub-classes in the importing code!*

Static Import

- It is the concept in which imported static members can be referred as if they belonged to the class that uses them
- It is not necessary for the class using them to refer them using their class name and dot operator
- Importing a particular member:

  import static packageName.ClassName.memberName;
- Importing all static members:

  import static packageName.ClassName.;*

References

1. https://www.startertutorials.com/corejava/access-control.html#:~:text=Access%20control%20is%20a%20mechanism,four%20access%20modifiers%20in%20Java.
2. https://www.tutorialspoint.com/can-we-declare-a-top-level-class-as-protected-or-private-in-java#:~:text=No%2C%20we%20cannot%20declare%20a,or%20default%20(no%20modifier).
3. https://www.javatpoint.com/access-modifiers
4.
5.