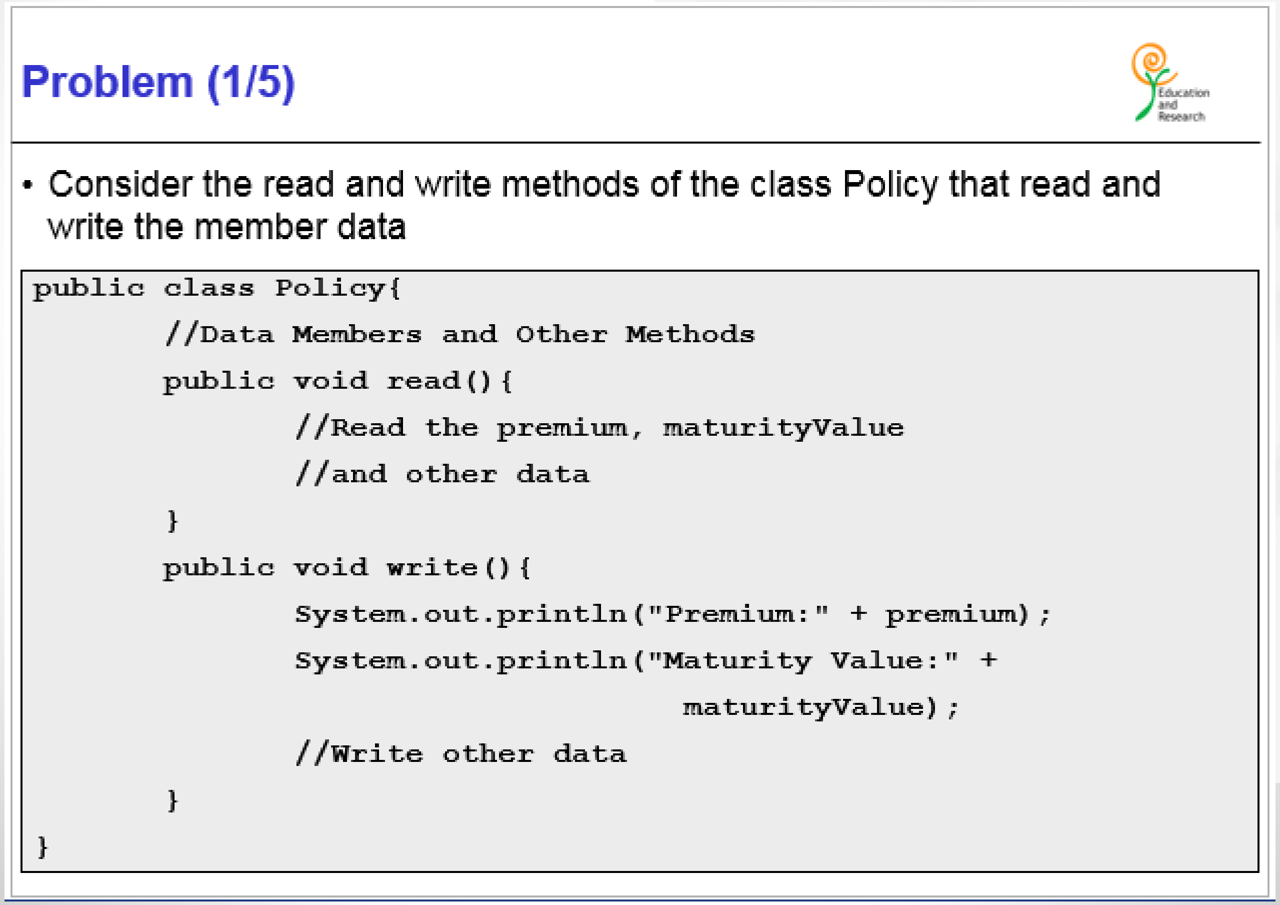
ITMD 411

Abstraction

ILLINOIS INSTITUTE OF TECHNOLOGY

Transforming Lives.Inventing the Future.www.iit.edu

ITMD 411 Key word - extends



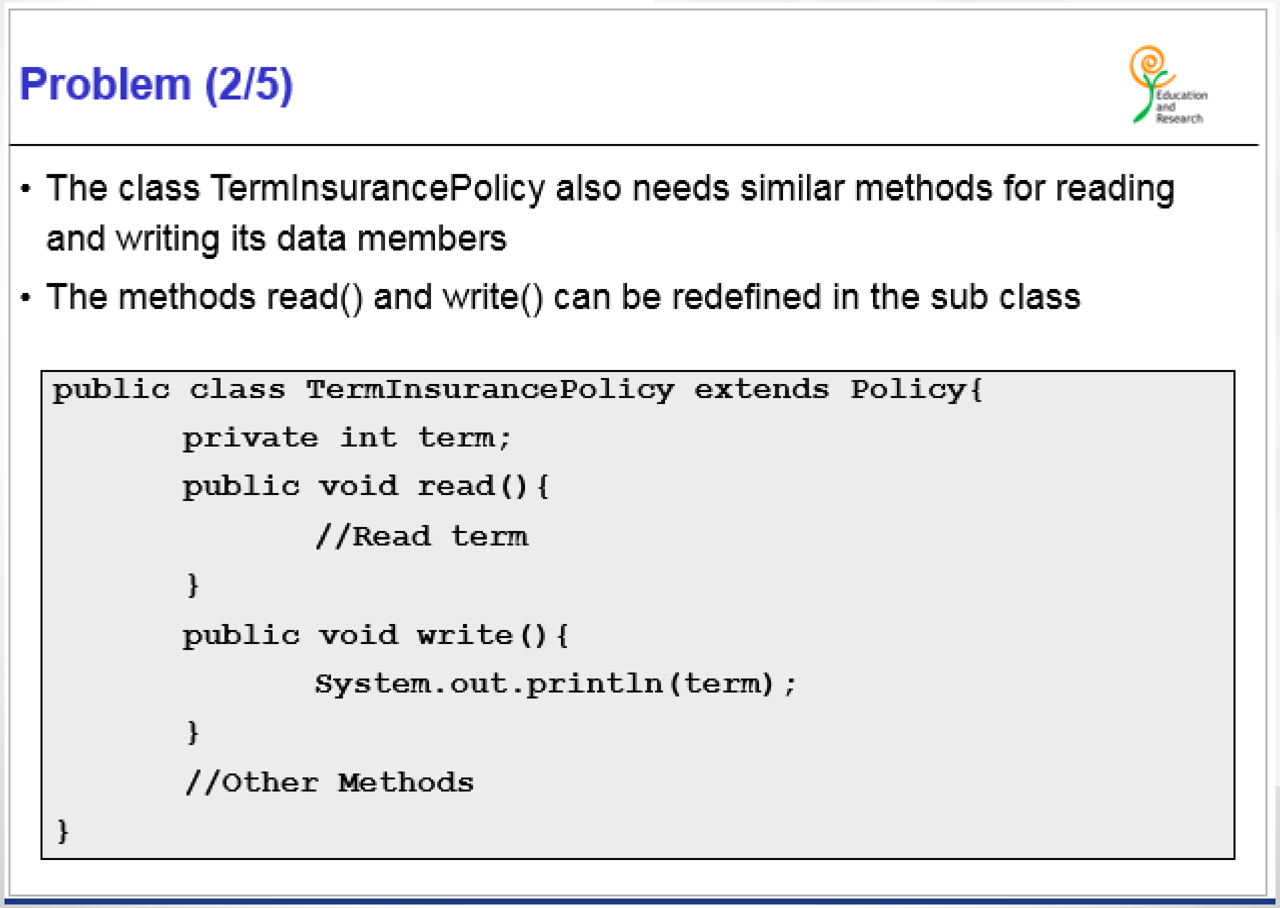
Problem (1/5)

Education and Research

- Consider the read and write methods of the class Policy that read and write the member data

```
public class Policy{
        //Data Members and Other Methods
        public void read(){
                //Read the premium, maturityValue
                //and other data
        }
        public void write(){
                System.out.println("Premium:" + premium);
                System.out.println("Maturity Value:" +
                                            maturityValue);
                //Write other data
        }
}
```

ITMD 411

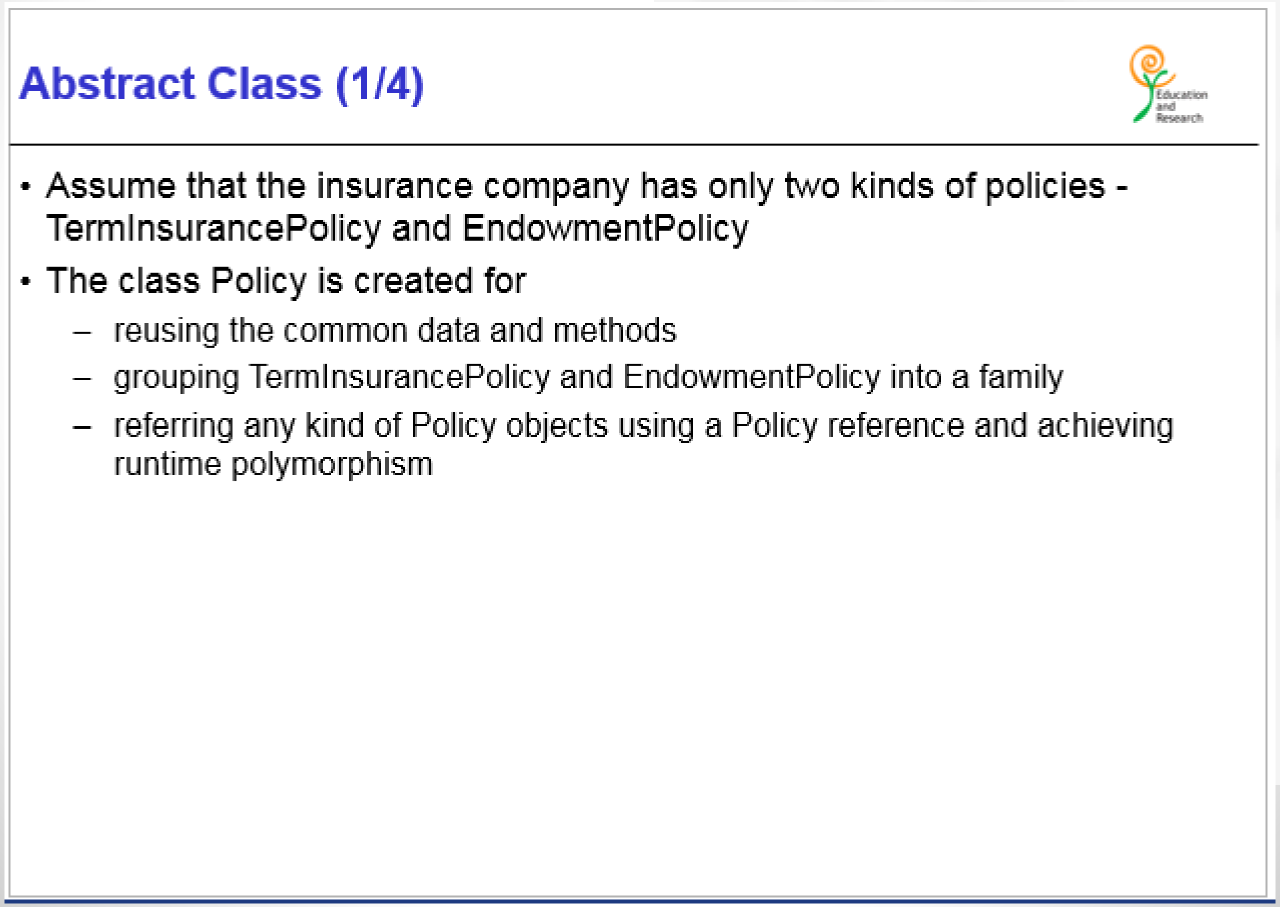


Problem (2/5)

- The class TermInsurancePolicy also needs similar methods for reading and writing its data members
- The methods read() and write() can be redefined in the sub class

```
public class TermInsurancePolicy extends Policy{
        private int term;
        public void read(){
                //Read term
        }
        public void write(){
                System.out.println(term);
        }
        //Other Methods
}
```

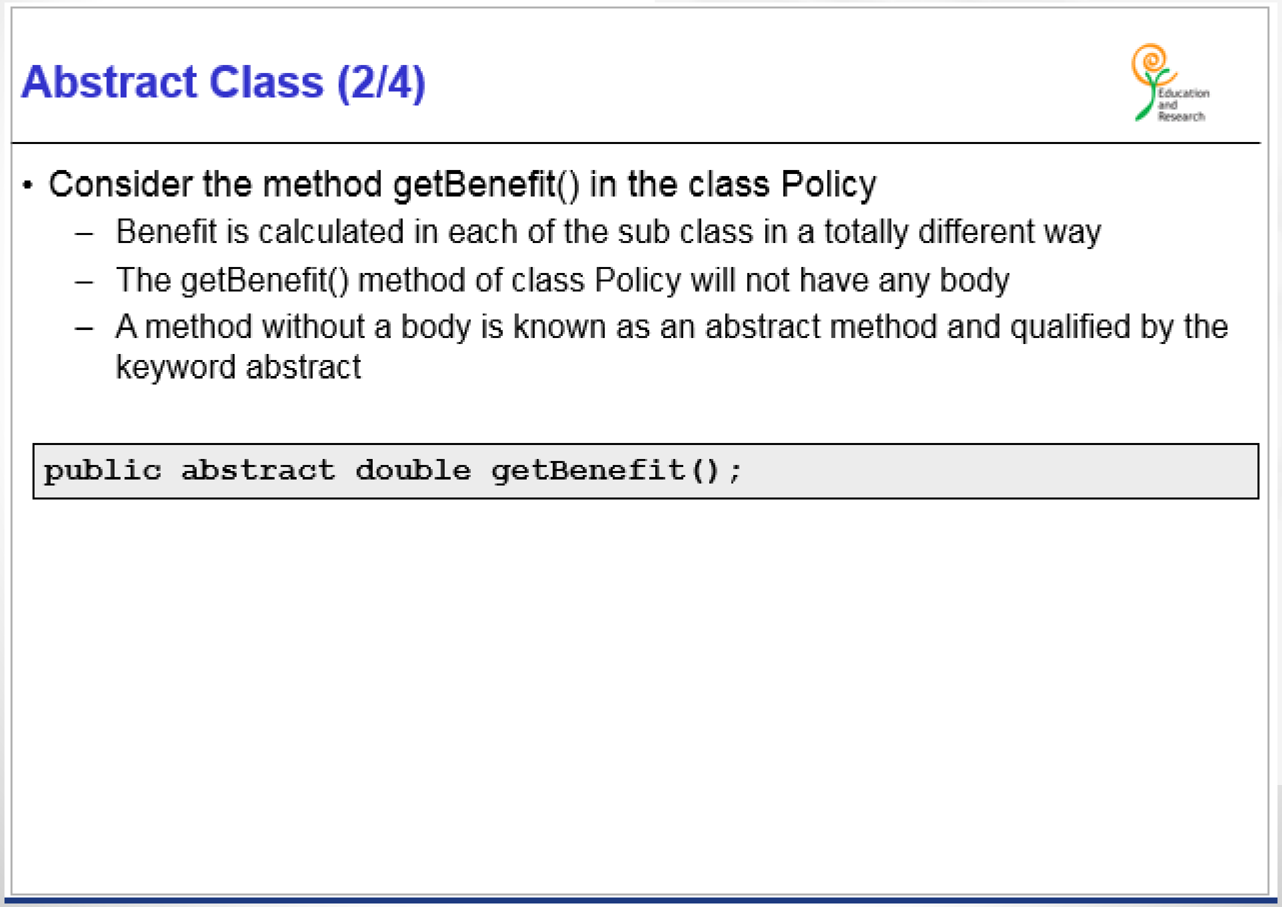
ITMD 411



Abstract Class (1/4)

- Assume that the insurance company has only two kinds of policies - TermInsurancePolicy and EndowmentPolicy
- The class Policy is created for
  - reusing the common data and methods
  - grouping TermInsurancePolicy and EndowmentPolicy into a family
  - referring any kind of Policy objects using a Policy reference and achieving runtime polymorphism

ITMD 411

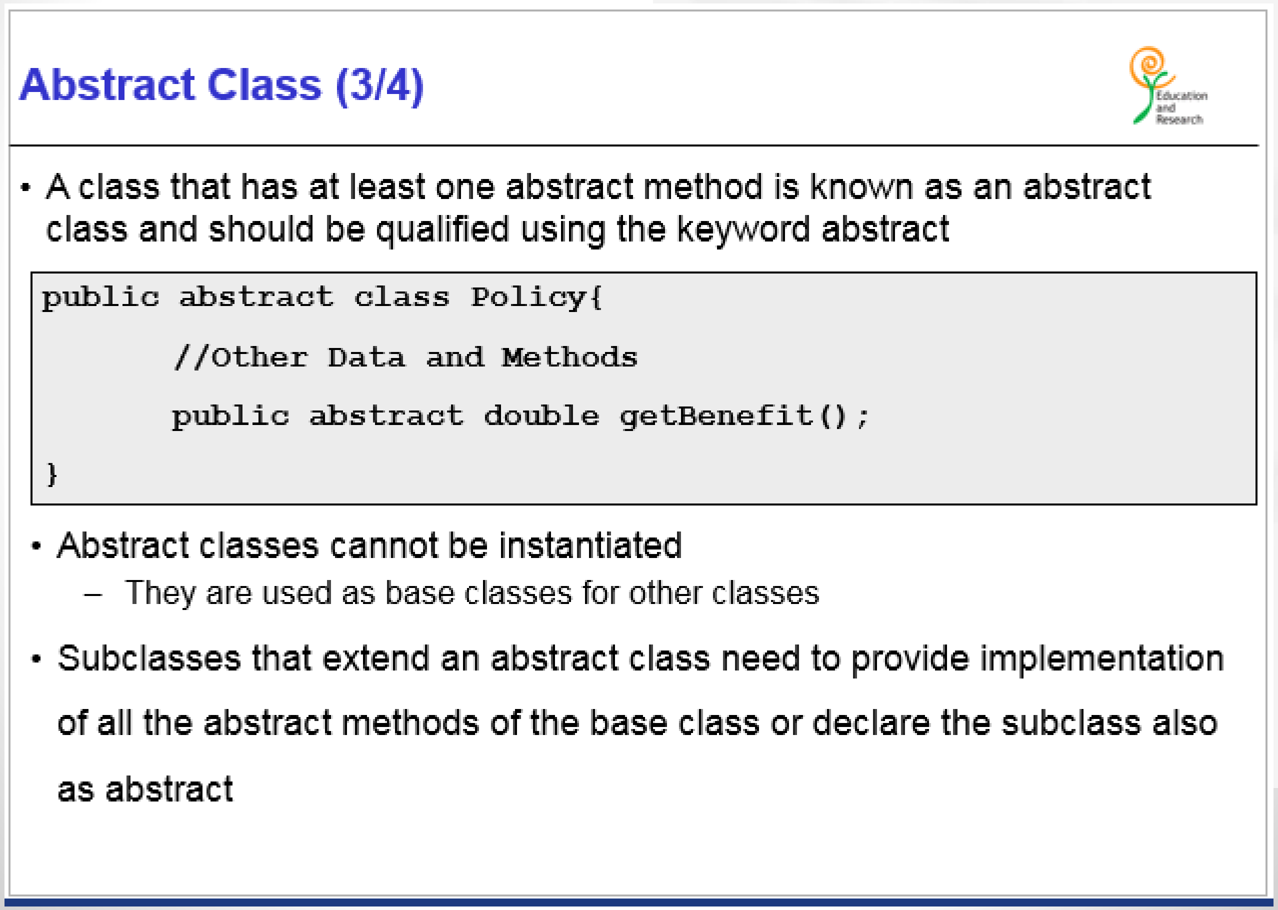


Abstract Class (2/4)

- Consider the method getBenefit() in the class Policy
  - Benefit is calculated in each of the sub class in a totally different way
  - The getBenefit() method of class Policy will not have any body
  - A method without a body is known as an abstract method and qualified by the keyword abstract

```
public abstract double getBenefit();
```

ITMD 411



Abstract Class (3/4)

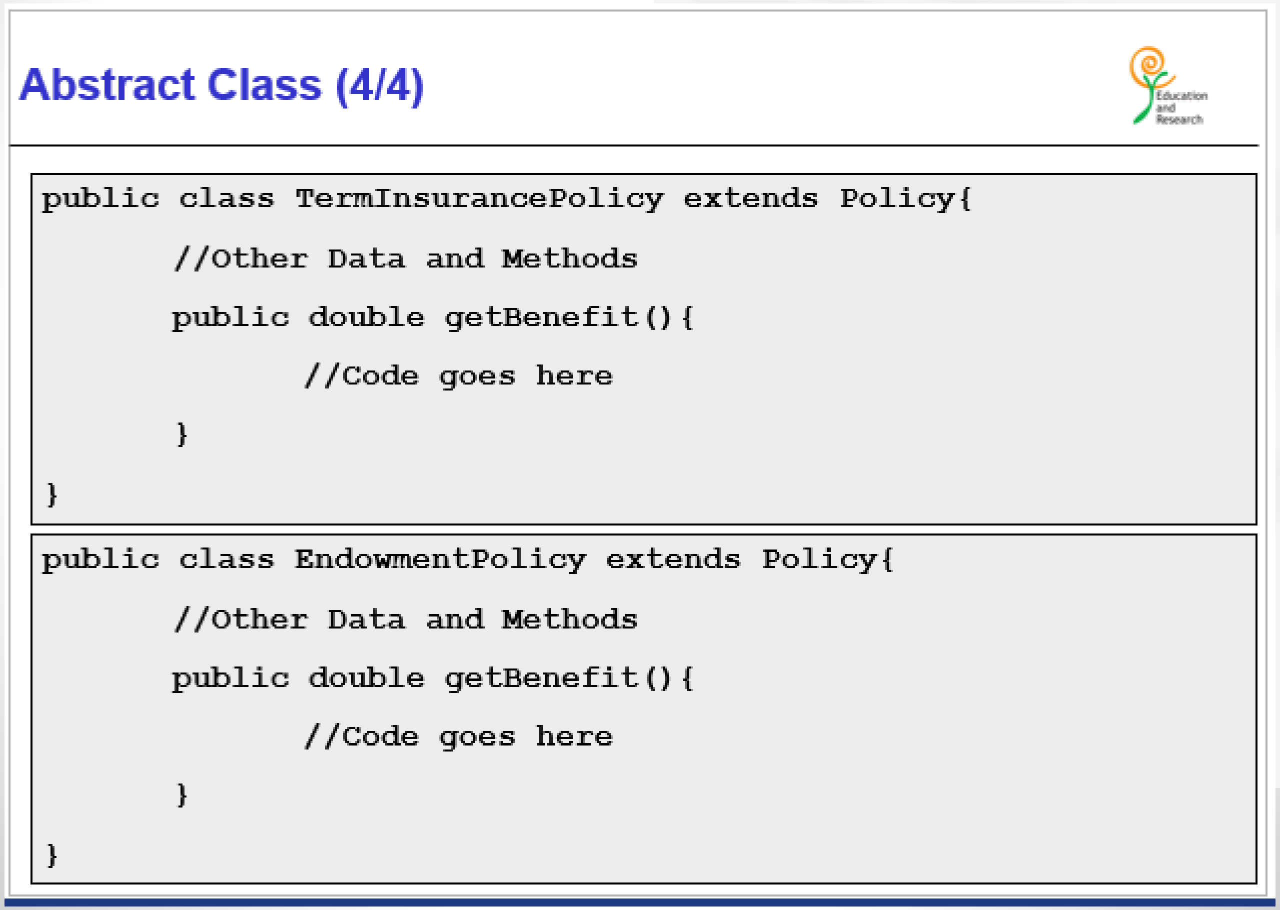
Education and Research

- A class that has at least one abstract method is known as an abstract class and should be qualified using the keyword abstract

```
public abstract class Policy{
       //Other Data and Methods
       public abstract double getBenefit();
}
```

- Abstract classes cannot be instantiated
  - They are used as base classes for other classes
- Subclasses that extend an abstract class need to provide implementation of all the abstract methods of the base class or declare the subclass also as abstract

ITMD 411

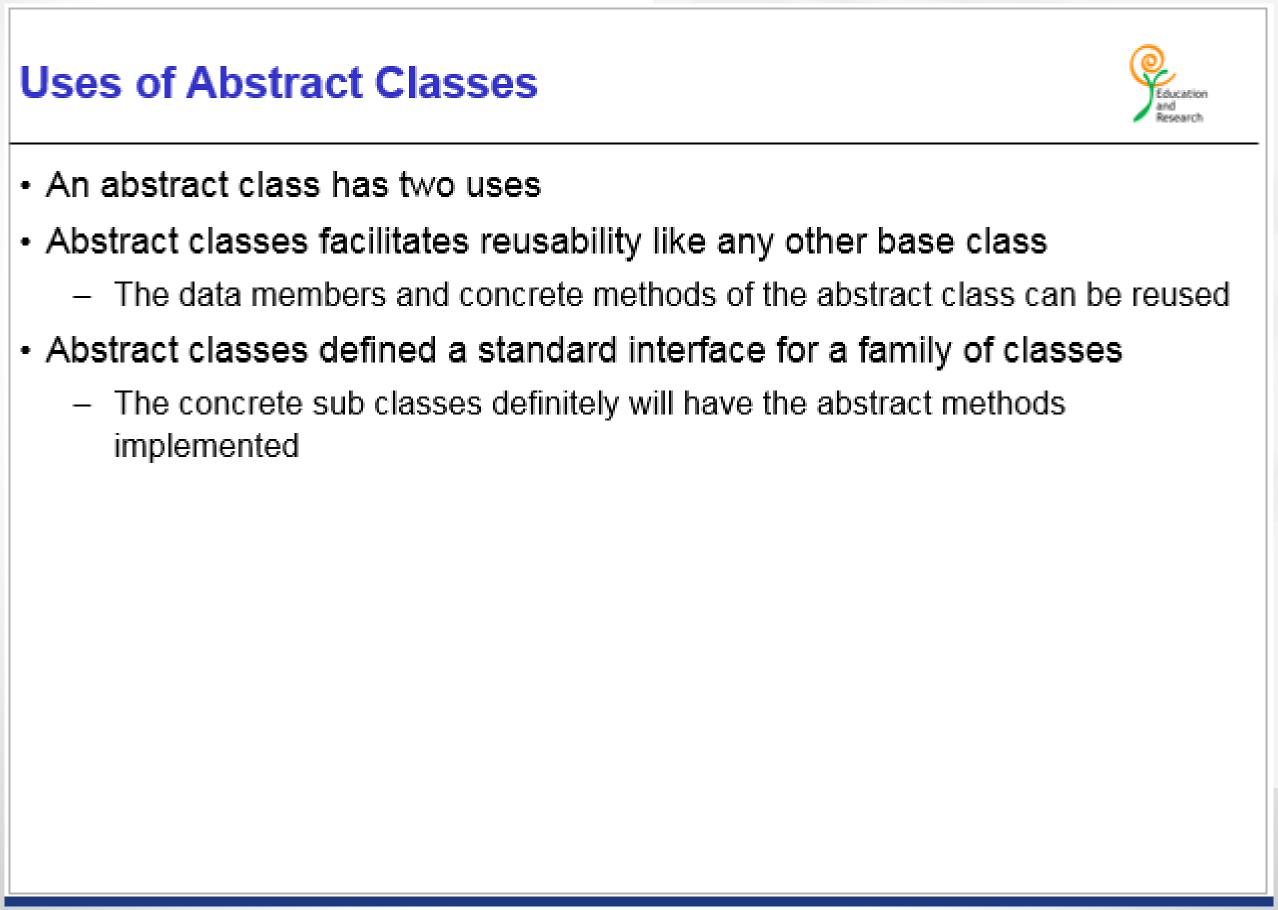


Abstract Class (4/4)

```
public class TermInsurancePolicy extends Policy{
        //Other Data and Methods
        public double getBenefit(){
                //Code goes here
        }
}
```

```
public class EndowmentPolicy extends Policy{
        //Other Data and Methods
        public double getBenefit(){
                //Code goes here
        }
}
```

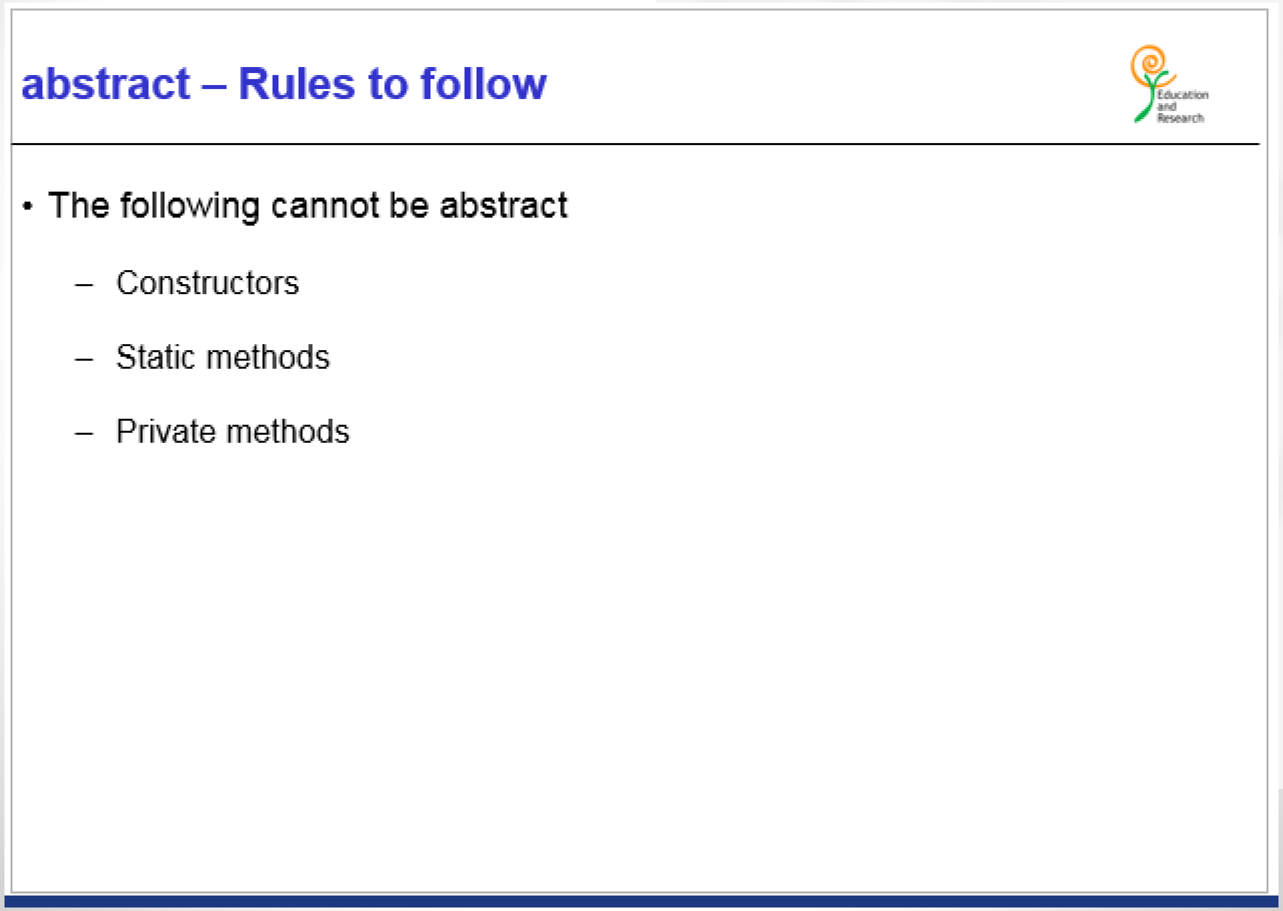
ITMD 411



Uses of Abstract Classes

- An abstract class has two uses
- Abstract classes facilitates reusability like any other base class
  - The data members and concrete methods of the abstract class can be reused
- Abstract classes defined a standard interface for a family of classes
  - The concrete sub classes definitely will have the abstract methods implemented

ITMD 411

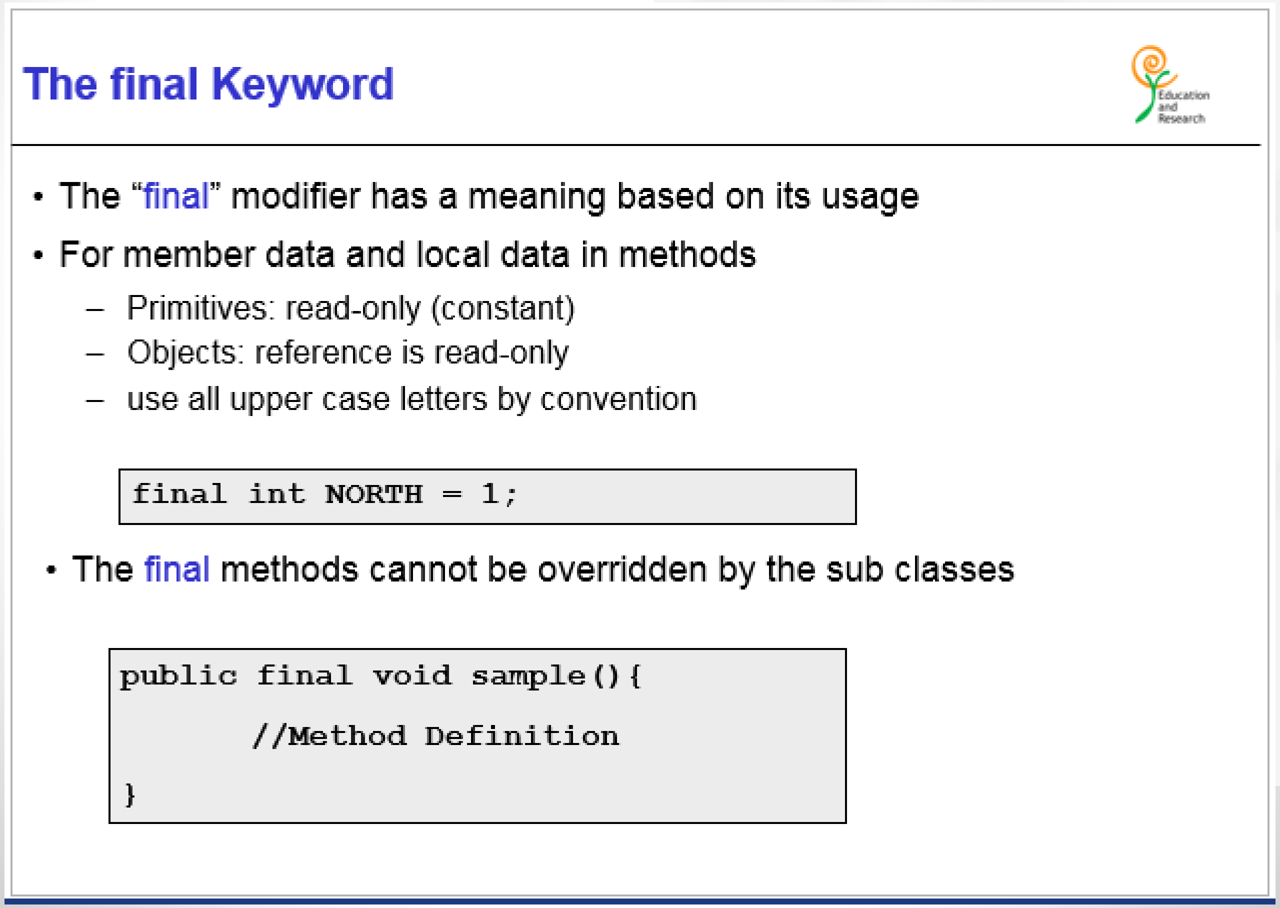


abstract – Rules to follow

Education and Research

- The following cannot be abstract
  - Constructors
  - Static methods
  - Private methods

ITMD 411



The final Keyword

- The “final” modifier has a meaning based on its usage
- For member data and local data in methods
  - Primitives: read-only (constant)
  - Objects: reference is read-only
  - use all upper case letters by convention

```
final int NORTH = 1;
```

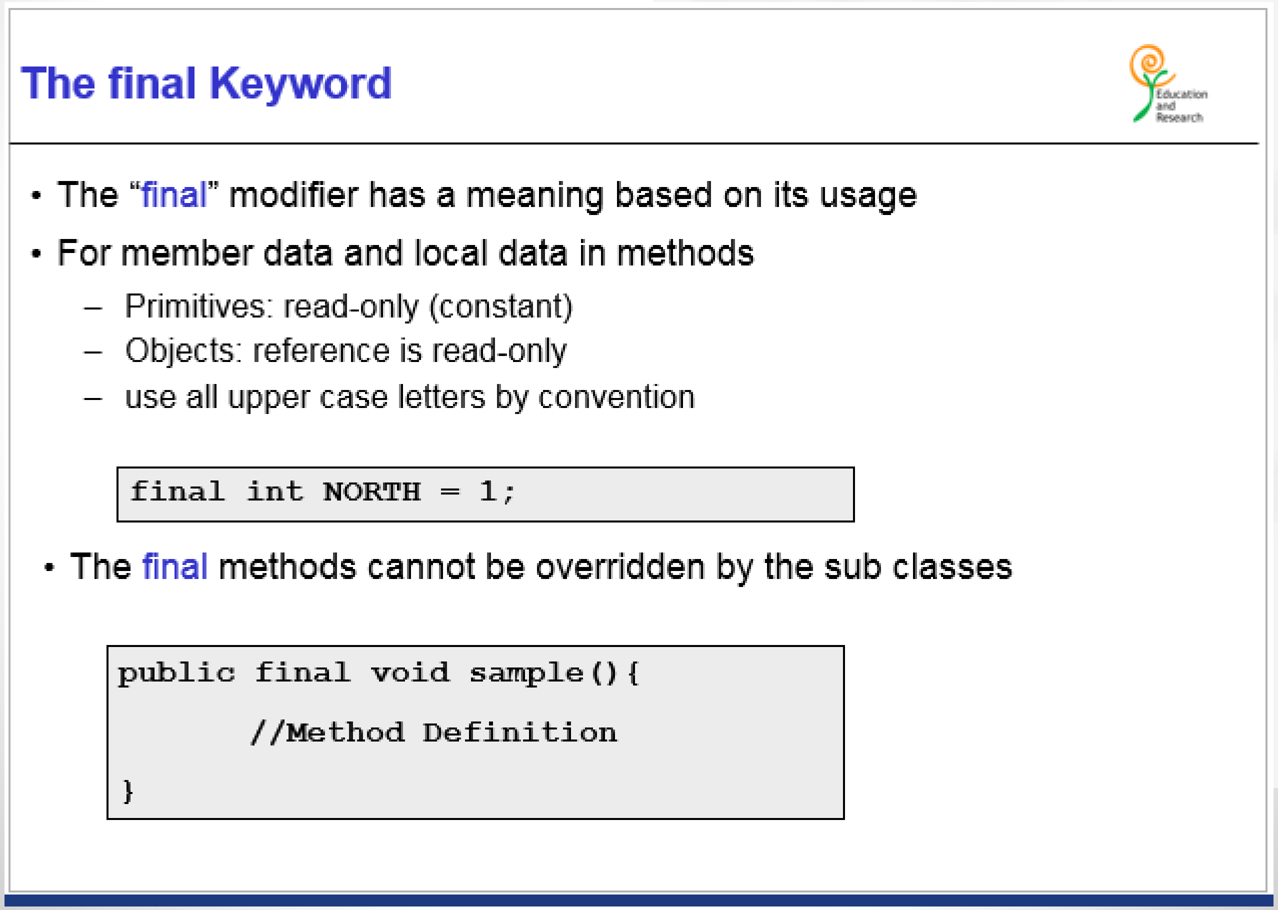
- The final methods cannot be overridden by the sub classes

```
public final void sample(){

       //Method Definition

}
```

ITMD 411



The final Keyword

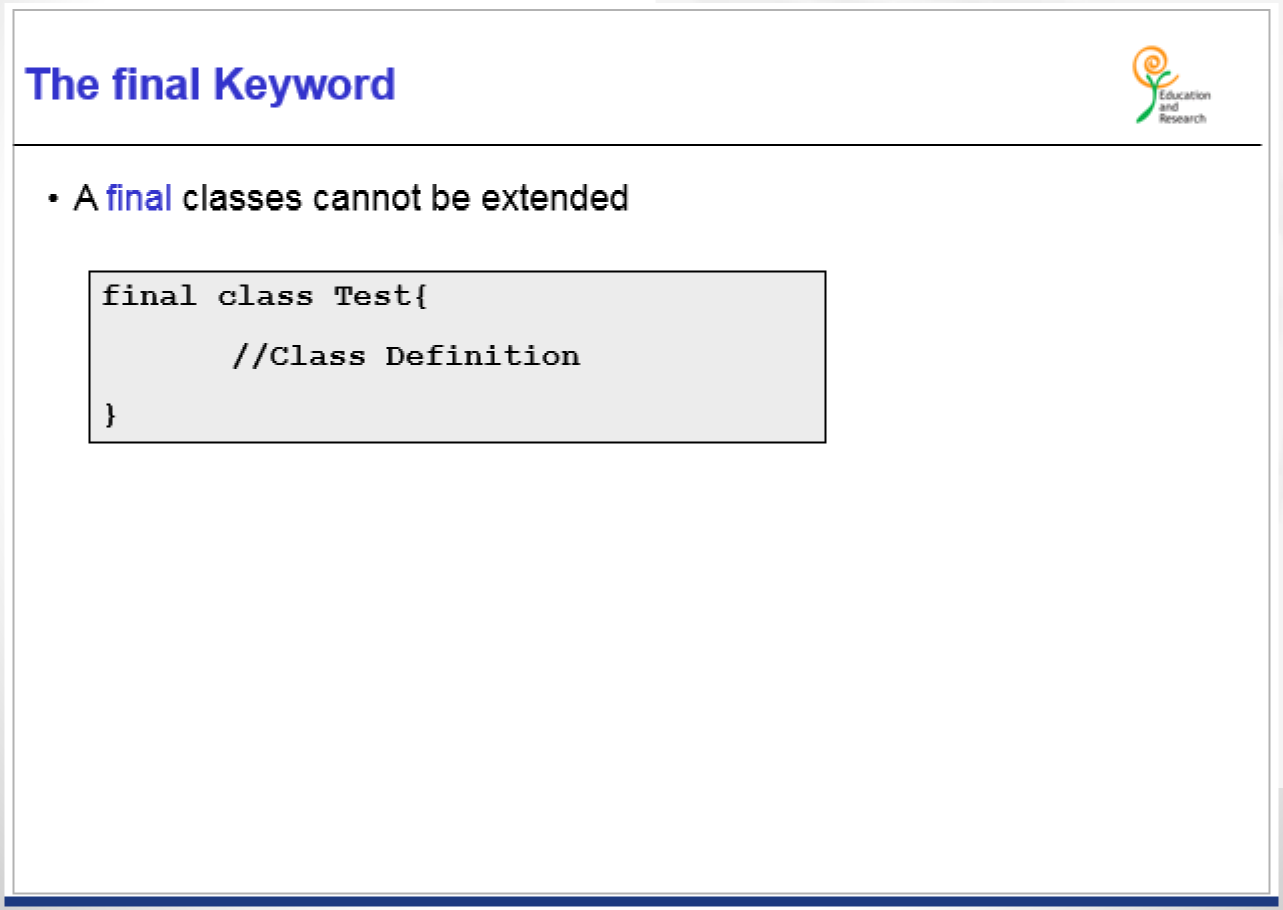
- The “final” modifier has a meaning based on its usage
- For member data and local data in methods
  - Primitives: read-only (constant)
  - Objects: reference is read-only
  - use all upper case letters by convention

```
final int NORTH = 1;
```

- The final methods cannot be overridden by the sub classes

```
public final void sample(){
        //Method Definition
}
```

ITMD 411



The final Keyword

- A final classes cannot be extended

```
final class Test{
        //Class Definition
}
```

Questions ??????????