Subclasses and inheritance

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming Concepts using Java

Week 3

A Java class

- An `Employee` class

```
public class Employee{
  private String name;
  private double salary;

  // Some Constructors ...

  // "mutator" methods
  public boolean setName(String s){ ... }
  public boolean setSalary(double x){ ... }

  // "accessor" methods
  public String getName(){ ... }
  public double getSalary(){ ... }

  // other methods
  public double bonus(float percent){
     return (percent/100.0)*salary;
  }
}
```

A Java class

- An `Employee` class
- Two private instance variables

```
public class Employee{
  private String name;
  private double salary;

  // Some Constructors ...

  // "mutator" methods
  public boolean setName(String s){ ... }
  public boolean setSalary(double x){ ... }

  // "accessor" methods
  public String getName(){ ... }
  public double getSalary(){ ... }

  // other methods
  public double bonus(float percent){
     return (percent/100.0)*salary;
  }
}
```

A Java class

- An `Employee` class
- Two private instance variables
- Some constructors to set up the object

```
public class Employee{
  private String name;
  private double salary;

  // Some Constructors ...

  // "mutator" methods
  public boolean setName(String s){ ... }
  public boolean setSalary(double x){ ... }

  // "accessor" methods
  public String getName(){ ... }
  public double getSalary(){ ... }

  // other methods
  public double bonus(float percent){
     return (percent/100.0)*salary;
  }
}
```

A Java class

- An `Employee` class
- Two private instance variables
- Some constructors to set up the object
- Accessor and mutator methods to set instance variables

```
public class Employee{
  private String name;
  private double salary;

  // Some Constructors ...

  // "mutator" methods
  public boolean setName(String s){ ... }
  public boolean setSalary(double x){ ... }

  // "accessor" methods
  public String getName(){ ... }
  public double getSalary(){ ... }

  // other methods
  public double bonus(float percent){
     return (percent/100.0)*salary;
  }
}
```

A Java class

- An `Employee` class
- Two private instance variables
- Some constructors to set up the object
- Accessor and mutator methods to set instance variables
- A public method to compute bonus

```
public class Employee{
  private String name;
  private double salary;

  // Some Constructors ...

  // "mutator" methods
  public boolean setName(String s){ ... }
  public boolean setSalary(double x){ ... }

  // "accessor" methods
  public String getName(){ ... }
  public double getSalary(){ ... }

  // other methods
  public double bonus(float percent){
     return (percent/100.0)*salary;
  }
}
```

Subclasses

- Managers are special types of employees with extra features

```
public class Manager extends Employee{
     private String secretary;
     public boolean setSecretary(name s){ ... }
     public String getSecretary(){ ... }
}
```

Subclasses

- Managers are special types of employees with extra features

```
public class Manager extends Employee{
     private String secretary;
     public boolean setSecretary(name s){ ... }
     public String getSecretary(){ ... }
}
```

- `Manager` objects inherit other fields and methods from `Employee`
  - Every `Manager` has a `name`, `salary` and methods to access and manipulate these.

Subclasses

- Managers are special types of employees with extra features

```
public class Manager extends Employee{
     private String secretary;
     public boolean setSecretary(name s){ ... }
     public String getSecretary(){ ... }
}
```

- `Manager` objects inherit other fields and methods from `Employee`
  - Every `Manager` has a `name`, `salary` and methods to access and manipulate these.
- `Manager` is a subclass of `Employee`
  - Think of subset

Subclasses

- `Manager` objects do not automatically have access to private data of parent class.
    - Common to extend a parent class written by someone else

Subclasses

- `Manager` objects do not automatically have access to private data of parent class.
    - Common to extend a parent class written by someone else
- How can a constructor for `Manager` set instance variables that are private to `Employee`?

Subclasses

- `Manager` objects do not automatically have access to private data of parent class.
  - Common to extend a parent class written by someone else
- How can a constructor for `Manager` set instance variables that are private to `Employee`?
- Some constructors for `Employee`

```
public class Employee{
  ...
  public Employee(String n, double s){
     name = n; salary = s;
  }
  public Employee(String n){
     this(n,500.00);
  }
}
```

Subclasses

- `Manager` objects do not automatically have access to private data of parent class.
    - Common to extend a parent class written by someone else
- How can a constructor for `Manager` set instance variables that are private to `Employee`?
- Some constructors for `Employee`
- Use parent class's constructor using `super`

```
public class Employee{
  ...
  public Employee(String n, double s){
     name = n; salary = s;
  }
  public Employee(String n){
     this(n,500.00);
  }
}
```

Subclasses

- Manager objects do not automatically have access to private data of parent class.
  - Common to extend a parent class written by someone else
- How can a constructor for Manager set instance variables that are private to Employee?
- Some constructors for Employee
- Use parent class's constructor using super
- A constructor for Manager

```
public class Employee{
  ...
  public Employee(String n, double s){
     name = n; salary = s;
  }
  public Employee(String n){
     this(n,500.00);
  }
}
```

```
public class Manager extends Employee{
  ..
  public Manager(String n, double s, String sn){
     super(n,s);   /* super calls
                      Employee constructor */
     secretary = sn;
  }
}
```

Inheritance

- In general, subclass has more features than parent class
  - Subclass inherits instance variables, methods from parent class

Inheritance

- In general, subclass has more features than parent class
  - Subclass inherits instance variables, methods from parent class
- Every `Manager` is an `Employee`, but not vice versa!

Inheritance

- In general, subclass has more features than parent class
  - Subclass inherits instance variables, methods from parent class
- Every `Manager` is an `Employee`, but not vice versa!
- Can use a subclass in place of a superclass

  ```
  Employee e = new Manager(...)
  ```

Inheritance

- In general, subclass has more features than parent class
  - Subclass inherits instance variables, methods from parent class
- Every `Manager` is an `Employee`, but not vice versa!
- Can use a subclass in place of a superclass

  ```
  Employee e = new Manager(...)
  ```
- But the following will not work

  ```
  Manager m = new Employee(...)
  ```

Inheritance

- In general, subclass has more features than parent class
  - Subclass inherits instance variables, methods from parent class
- Every `Manager` is an `Employee`, but not vice versa!
- Can use a subclass in place of a superclass

  `Employee e = new Manager(...)`
- But the following will not work

  `Manager m = new Employee(...)`

- Recall
  - `int[] a = new int[100];`
  - Why the seemingly redundant reference to `int` in `new`?

Inheritance

- In general, subclass has more features than parent class
    - Subclass inherits instance variables, methods from parent class
- Every `Manager` is an `Employee`, but not vice versa!
- Can use a subclass in place of a superclass

  `Employee e = new Manager(...)`
- But the following will not work

  `Manager m = new Employee(...)`
- Recall
    - `int[] a = new int[100];`
    - Why the seemingly redundant reference to `int` in `new`?
- One can now presumably write

  `Employee[] e = new Manager(...)[100]`

Summary

- A subclass extends a parent class
- Subclass inherits instance variables and methods from the parent class
- Subclass can add more instance variables and methods
  - Can also override methods — later
- Subclasses cannot see private components of parent class
- Use `super` to access constructor of parent class