- **Interfaces**

- **Interfaces vs. Abstract classes**

- **What is inheritance with respect to interfaces ?**

Interface

- collection of abstract methods
- it may have constants
- methods are implicitly public, abstract
- constants are implicitly public, static, final
- a class can implement one or more interfaces

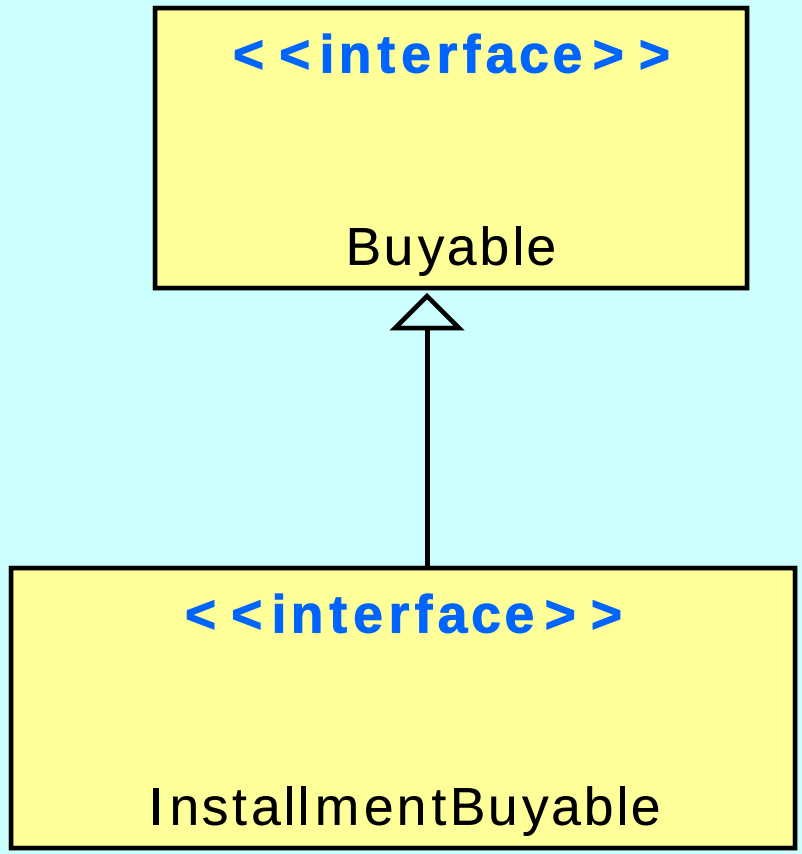
```
public interface Pet {
   void play();
}

public class Dog extends Canine implements
Pet {
   public void play() {
      ……
   }
}
```

Interface versus Abstract class

- An interface can have only abstract methods. An abstract class can have non-abstract methods. (An abstract class does not have to contain abstract methods).

- An interface can have constants. An abstract class can have constants, variables.

Interface hierarchy



Visibility issues ??

inherits all abstract methods and constants of parent

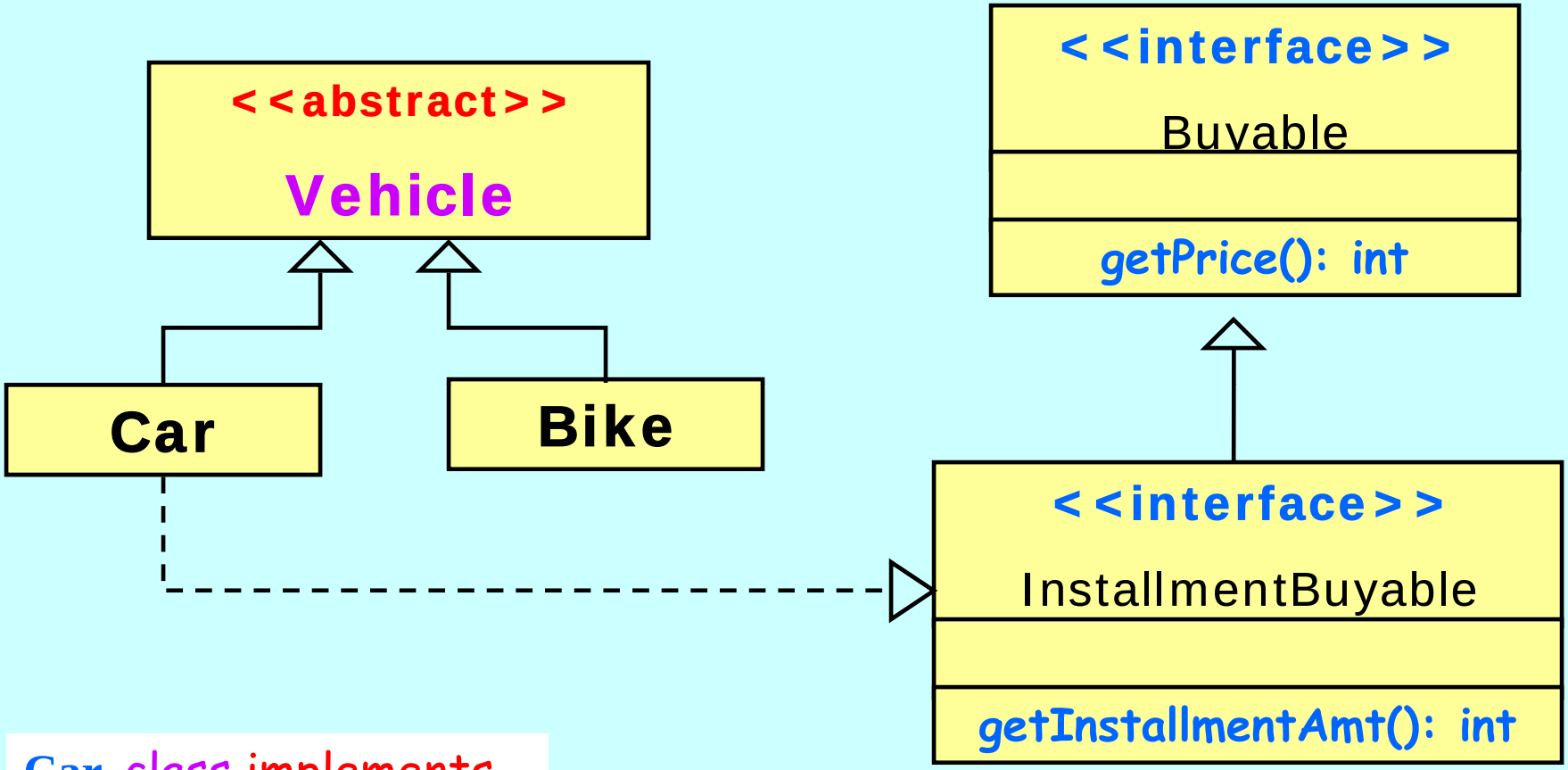
Interface: Example

```
public interface Buyable
{
      String  getPrice();
}
```

Interface: Example

```
public interface InstallmentBuyable extends Buyable
{
    int ADMIN_FEE = 100;
    int getInstallmentAmt();
}
```

Design showing class implements interface



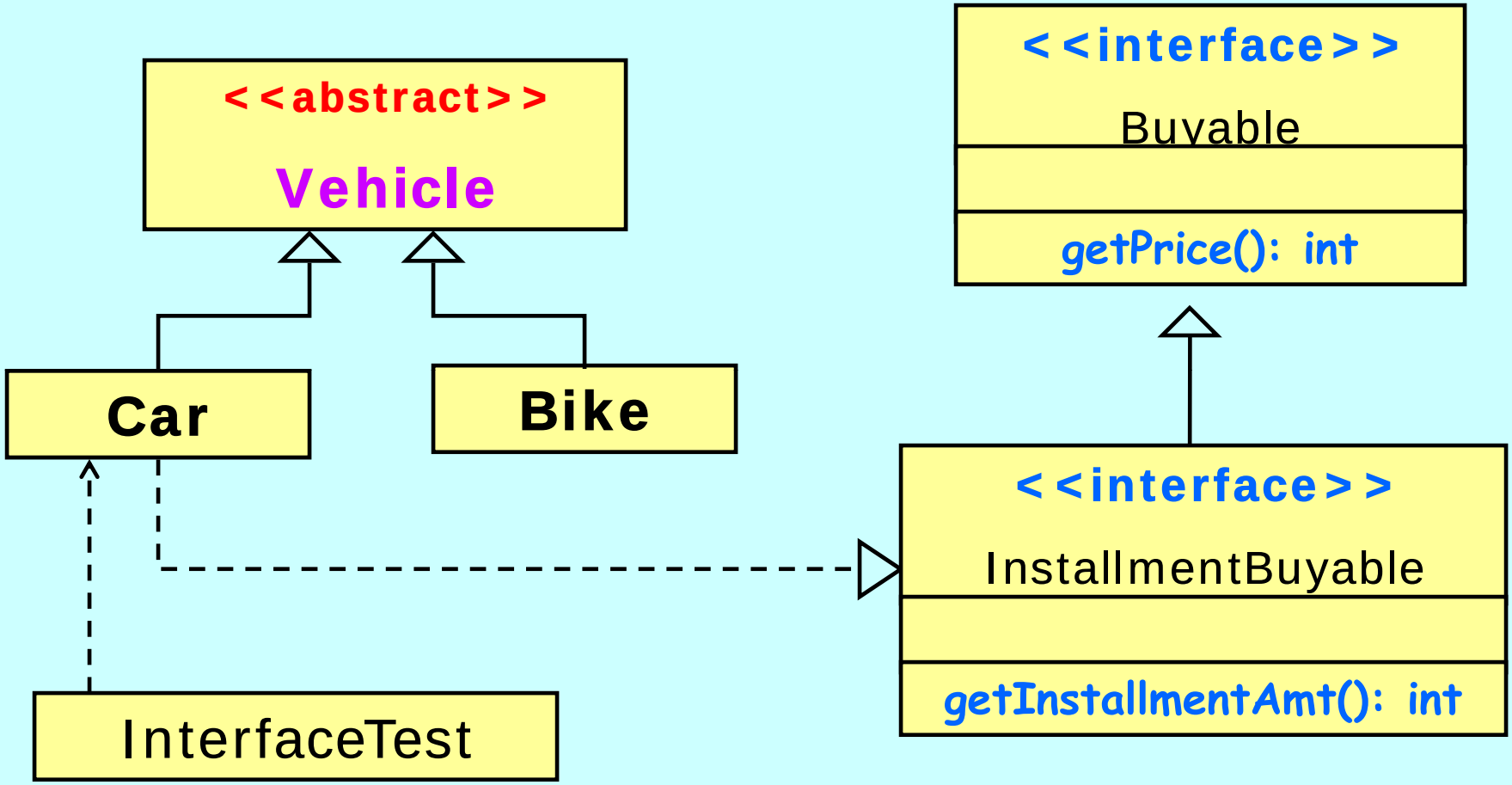
Car class implements InstallmentBuyable interface

"implements" relationship

Car implements interface InstallmentBuyable

```
public class Car extends Vehicle
                 implements InstallmentBuyable {

  private int period, price;
  public Car( int pe, int pr ) {
     period = pe; price = pr;
  }

  public boolean getInstallmentAmt() {
    return ( ADMIN_FEE + price )/ period;
  }
  public String getPrice() { return price ; }
}
```

Design for our interface testing



Interface Usage Testing

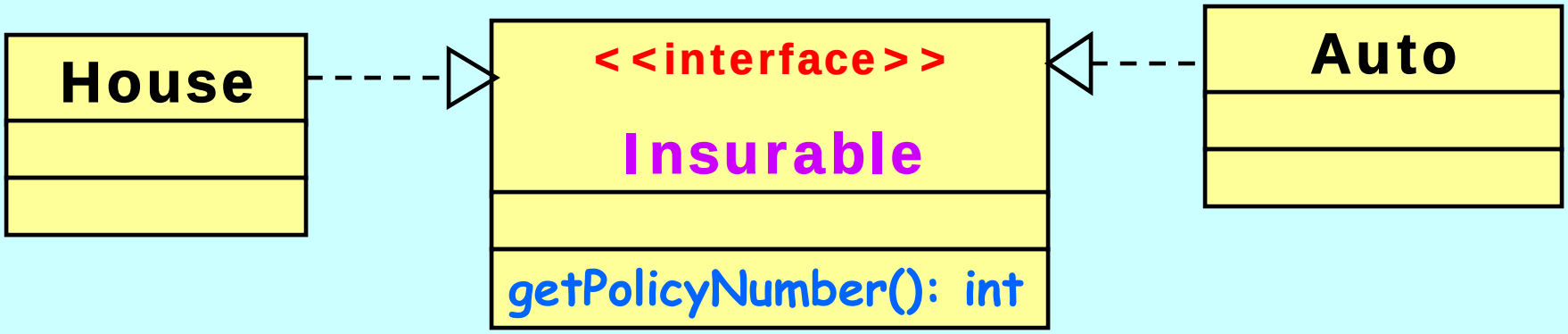
```
public class InterfaceTest {
   public static void main( String[] args ) {

      Car  c = new Car( 12, 20000 );

      System.out.println( c.getPrice() );

      System.out.println( c.getInstallmentAmt() );
   }
}
```

output ?

Polymorphism (via interfaces)

```
Insurable i = new House();
i = new Auto();
```

- Lets see an example:



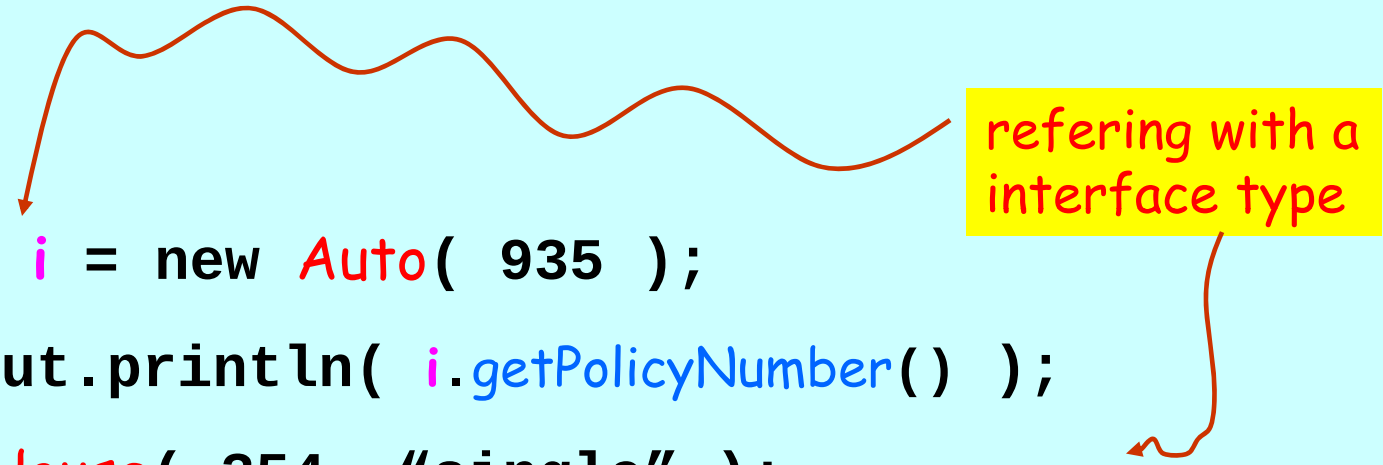
```
public interface Insurable {
    int getPolicyNumber();
}
```

```
public class Auto implements Insurable {
    private int policy;
    public Auto( int p ) { policy = p; }
    public int getPolicyNumber() {return policy;}
}
```

```
public class House implements Insurable {
  private int policy;
  private String type;

  public House( int p, String t ) {
      policy = p;
      type = t;
  }
  public int getPolicyNumber() {return policy;}

  public String getType() { return type; }

}
```

Polymorphism (via interface) test – Example

```
public class PolyTest {
   public static void main( String[] args )
   {
      Insurable i = new Auto( 935 );
      System.out.println( i.getPolicyNumber() );
      i = new House( 354, "single" );
      System.out.println( i.getPolicyNumber() );
   }
}
```



outputs ??

```
public class PolyTest {
   public static void main( String[] args )
   {
      Insurable h = new House( 354, “single” );

      String type = ((House)h).getType();
   }
}
```

compile ??