The Last of the Producer App

Plan for This Week

- Today
  - The last of the Payroll App.
  - Project 1
- Tomorrow and Wednesday:
  - Java I/O

Plan ahead…

- The week we return:
  - Wednesday, Jan 7th – Exam 1
    - Will cover
      - Inheritance
      - Interfaces
      - Exceptions
  - Friday, Jan 9th
    - Project 1 – minimum submission

Before we start

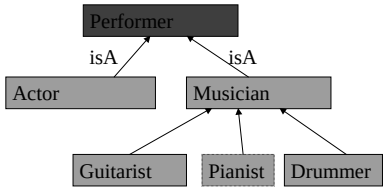
- Any questions?

Back to our Payroll app

- Final code and javadocs on my Web site.

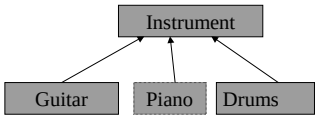
Back to our Payroll app

- Class hierarchy for theatre app



Back to our Payroll app

- Instrument hierarchy



Changes to the Payroll class

```
public class Payroll {
private Performer performer[];
private int nPerf;

public void addPerformer (Performer P) throws
   PayrollFullException
{
   if (nPerf == MAXPERF)
       throw new PayrollFullException()

   performer[nPerf] = P;
   nPerf++;
}
```

calculateTotalPay

```
public double calculateTotalPay()
{
    double sum = 0.0;
    for (int i=0; i < nPerf; i++)
       sum += performer[i].calculatePay();
    return sum;
}
```

A look at Instruments

```
public abstract class Instrument {
private double rentalCost;

protected Instrument (double cost)
{
   rentalCost = cost;
}

public double getWeeklyRental ()
{
   return rentalCost;
}
}
```

A look at instruments

```
public class Guitar extends Instrument {

public Guitar ()
{
   super (200.0);
}

protected Guitar (double rate)
{
   super (rate);
}

}
```

A look at Performer

```
public abstract class Performer implements Comparable {
private String myName;
private double payPerPerf;
private int nPerformances;

protected Performer (String name, double rate)
{
   myName = name;
   payPerPerf = rate;
   nPerformances = 0;
}

}
```

A look at Performer

```
public abstract double calculatePay();

protected double getBasePay()
{
  return nPerformances * payPerPerf;
}

public void perform (int n)
{
  nPerformances = n;
}
```

Now Actor

```
public class Actor extends Performer {

private static final double PAYRATE = 200.0;

public Actor (String name)
{
  super (name, PAYRATE);
}

 public double calculatePay ()
{
  return getBasePay();
}

}
```

And Musician

```
public class Musician extends Performer {

private Instrument myInstrument;
private static final double PAYRATE = 100.0;

public Musician(String name, Instrument I)
{
  super(name, PAYRATE);
  myInstrument = I;
}
public double calculatePay()
{
  return getBasePay() + myInstrument.getWeeklyRental();
}
```

Finally Musician subclasses

```
public class Drummer extends Musician {

public Drummer (String name)
{
  super (name, new Drums());
}

}
```

Payroll App

- Any questions?