```
            System.out.print(getAgeWoman());
        }
}
```

В методе **setAge()** из-за генерации исключения происходит преждевременный выход из блока **try**, но до выхода из функции выполняется раздел **finally**. Метод **getAgeWoman()** завершает работу выполнением стоящего в блоке **try** оператора **return**, но и при этом перед выходом из метода выполняется код блока **finally**.

Собственные исключения

Разработчик может создать собственное исключение как подкласс класса **Exception** и затем использовать его при обработке ситуаций, не являющихся исключениями с точки зрения языка, но нарушающих логику вещей. Например, появление объектов типа **Human** (Человек) с отрицательным значением поля **age** (возраст).

```
/* пример # 5 : метод, вызывающий исключение, созданное программистом:
 RunnerLogic.java  */
package chapt08;

public class RunnerLogic {
     public static double salary(int coeff)
                                   throws SalaryException {
          double d;
          try {
               if((d = 10 - 100/coeff) < 0)
               throw new SalaryException("negative salary");
               else return d;
          } catch (ArithmeticException e) {
               throw new SalaryException("div by zero", e);
          }
     }
     public static void main(String[] args) {
          try {
               double res = salary(3); //или 0, или 71;
          } catch (SalaryException e) {
               System.err.println(e.toString());
               System.err.println(e.getHiddenException());
          }
     }
}
```

При невозможности вычислить значение генерируется объект **ArithmeticException**, обработчик которого, в свою очередь, генерирует исключение **SalaryException**, используемое в качестве собственного исключения. Он принимает два аргумента. Один из них – сообщение, которое может быть выведено в поток ошибок; другой – реальное исключение, которое привело к вызову нашего исключения. Этот код показывает, как можно сохранить другую ин-

формацию внутри пользовательского исключения. Преимущество этого сохранения состоит в том, что если вызываемый метод захочет узнать реальную причину вызова **SalaryException**, он всего лишь должен вызвать метод **getHiddenException()**. Это позволяет вызываемому методу решить, нужно ли работать со специфичным исключением или достаточно обработки **SalaryException**.

```
/* пример # 6 : собственное исключение: SalaryException.java */
package chapt08;

public class SalaryException extends Exception {
     private Exception _hidden;

     public SalaryException(String er) {
          super(er);
     }
     public SalaryException(String er, Exception e) {
          super(er);
          _hidden = e;
     }
     public Exception getHiddenException() {
          return _hidden;
     }
}
```

Разработчики программного обеспечения стремятся к высокому уровню повторного использования кода, поэтому они постарались предусмотреть и закодировать все возможные исключительные ситуации. Поэтому при реальном программировании можно вполне обойтись без создания собственных классов исключений.

Наследование и исключения

Создание сложных распределенных систем редко обходится без наследования и обработки исключений. Следует знать два правила для проверяемых исключений при наследовании:

- переопределяемый метод в подклассе не может содержать в инструкции **throws** исключений, не обрабатываемых в соответствующем методе суперкласса;
- конструктор подкласса должен включить в свой блок **throws** все классы исключений или их суперклассы из блока **throws** конструктора суперкласса, к которому он обращается при создании объекта.

Первое правило имеет непосредственное отношение к расширяемости приложения. Пусть при добавлении в цепочку наследования нового класса его полиморфный метод включил в блок **throws** «новое» проверяемое исключение. Тогда методы логики пріложения, принимающие объект нового класса в качестве параметра и вызывающие данный полиморфный метод, не готовы обрабатывать «новое» исключение, так как ранее в этом не было необходимости. Поэтому при попытке добавления «нового» checked-исключения в полиморфный метод компилятор выдает сообщение об ошибке.