```
// следующий класс невозможен
class BaseCourse extends ConstCourse {/*код*/}
```

Использование super и this

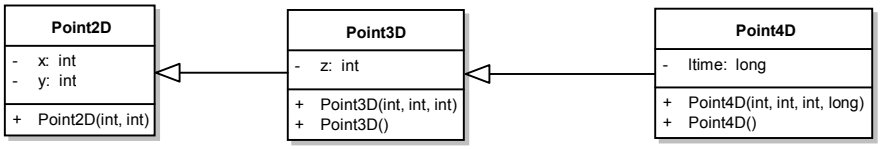
Ключевое слово **super** используется для вызова конструктора суперкласса и для доступа к члену суперкласса. Например:

```
super(список_параметров);/* вызов конструктора суперкласса
                            с передачей параметров или без нее*/
super.id = 71; /* обращение к атрибуту суперкласса */
super.getId(); // вызов метода суперкласса
```

Вторая форма **super** используется для доступа из подкласса к переменной **id** суперкласса. Третья форма специфична для Java и обеспечивает вызов из подкласса переопределенного метода суперкласса, причем если в суперклассе этот метод не определен, то будет осуществляться поиск по цепочке наследования до тех пор, пока метод не будет найден.

Каждый экземпляр класса имеет неявную ссылку **this** на себя, которая передается также и методам. После этого метод «знает», какой объект его вызвал. Вместо обращения к атрибуту **id** в методах можно писать **this.id**, хотя и не обязательно, так как записи **id** и **this.id** равносильны.

Следующий код показывает, как, используя **this**, можно строить одни конструкторы на основе других.




```
// пример # 3 : this в конструкторе: Point2D.java, Point3D.java, Point4D.java
package chapt04;

public class Point2D {
      private int x, y;

      public Point2D(int x, int y) {
            this.x = x;//this используется для присваивания полям класса
            this.y = y;//x, y, значений параметров конструктора x, y, z
      }
}
package chapt04;

public class Point3D extends Point2D {
      private int z;

      public Point3D(int x, int y, int z) {
            super(x, y);
            this.z = z;
      }
```

```
        public Point3D() {
            this(-1,-1,-1);// вызов конструктора Point3D с параметрами
        }
}
package chapt04;

public class Point4D extends Point3D{
        private long time;

        public Point4D(int x, int y, int z, long time) {
                super(x, y, z);
                this.time = time;
        }
        public Point4D() {
                // по умолчанию super();
        }
}
```

В классе **Point3D** второй конструктор для завершения инициализации объекта обращается к первому конструктору. Такая конструкция применяется в случае, когда в класс требуется добавить конструктор по умолчанию с обязательным использованием уже существующего конструктора.

Ссылка **this** используется в методе для уточнения того, о каких именно переменных **x**, **y** и **z** идет речь в методе, а конкретно для доступа к переменным класса из метода, если в методе есть локальные переменные с тем же именем, что и у класса. Инструкция **this()** должна быть единственной в вызывающем конструкторе и быть первой по счету выполняемой операцией.

Переопределение методов и полиморфизм

Способность Java делать выбор метода, исходя из типа объекта во время выполнения, называется ***поздним связыванием***. При вызове метода его поиск происходит сначала в данном классе, затем в суперклассе, пока метод не будет найден или не достигнут **Object** – суперкласс для всех классов.

Если два метода с одинаковыми именами и возвращаемыми значениями находятся в одном классе, то списки их параметров должны отличаться. То же относится к методам, наследуемым из суперкласса. Такие методы являются перегружаемыми (overloading). При обращении вызывается тот метод, список параметров которого совпадает со списком параметров вызова. Если объявление метода подкласса полностью, включая параметры, совпадает с объявлением метода суперкласса (порождающего класса), то метод подкласса переопределяет (overriding) метод суперкласса. Переопределение методов является основой концепции динамического связывания, реализующей полиморфизм. Когда переопределенный метод вызывается через ссылку суперкласса, Java определяет, какую версию метода вызвать, основываясь на типе объекта, на который имеется ссылка. Таким образом, тип объекта определяет версию метода на этапе выполнения. В следующем примере рассматривается реализация полиморфизма на основе динамического связывания. Так как суперкласс содержит методы, переопределенные подклассами, то объект суперкласса будет вызывать методы раз-