1. Наследование

В коде это пишется как `extends`, после которого указываете имя базового класса.

```
public class SubClass extends SuperClass { }
```

Теперь в классе `SubClass` нам доступны все `public/protected` поля и методы класса `SuperClass`.

Ключевое слово super

Ключевое слово `super`, которое обозначает суперкласс, т.е. класс, производным от которого является текущий класс.
`super` можно использовать для вызова конструктора суперкласса и для обращения к члену суперкласса, скрытому членом подкласса.

Вызов базового конструктора

```
class HeavyBox extends Box {
    int weight; // вес коробки

    // конструктор
    // инициализируем переменные с помощью ключевого слова super
    HeavyBox(int w, int h, int d, int m) {
        super(w, h, d); // вызов конструктора суперкласса
        weight = m; // масса
    }
}
```

> Вызов метода `super()` всегда должен быть первым оператором, выполняемым внутри конструктора подкласса.

Обращение к члену базового класса

```
super.example;
```

> Здесь `example` может быть методом либо переменной экземпляра.

Вызов конструкторов

В иерархии классов конструкторы вызываются в порядке наследования, начиная с суперкласса и заканчивая подклассом. Если метод `super()` не применяется, программа использует конструктор каждого суперкласса, заданный по умолчанию или не содержащий параметров.

Вы можете создать три класса A, B, C, которые наследуются друг от друга (A←B←C), у которых в конструкторе выводится текст и вызвать в основном классе код:

```
C c = new C();
```

> Вы должные увидеть три строчки текста.

Переопределение методов

Для того чтобы переопределить метод базового класса нужно в начале написать аннотацию: `@Override`

```
@Override
String sleep() {
    // TODO Auto-generated method stub
    return super.sleep();
}
```

Запрет наследования

Хотя наследование очень интересный и эффективный механизм, но в некоторых ситуациях его применение может быть нежелательным. И в этом случае можно запретить наследование с помощью ключевого слова `final`. Например:

```
public final class Person { }
```

Другое

Способность к изменению функциональности, унаследованной от базового класса, называется **полиморфизмом** и является одиним из ключевых аспектов объектно-ориентированного программирования наряду с наследованием и инкапсуляцией.