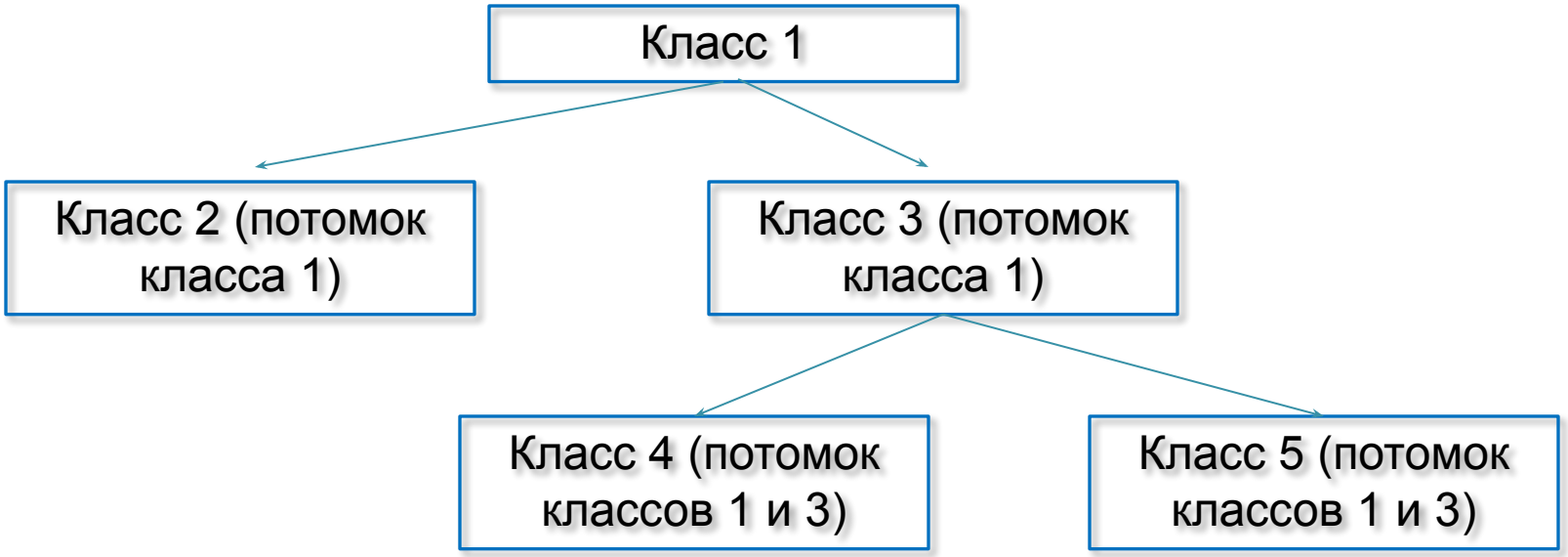
Тема 13

Наследование

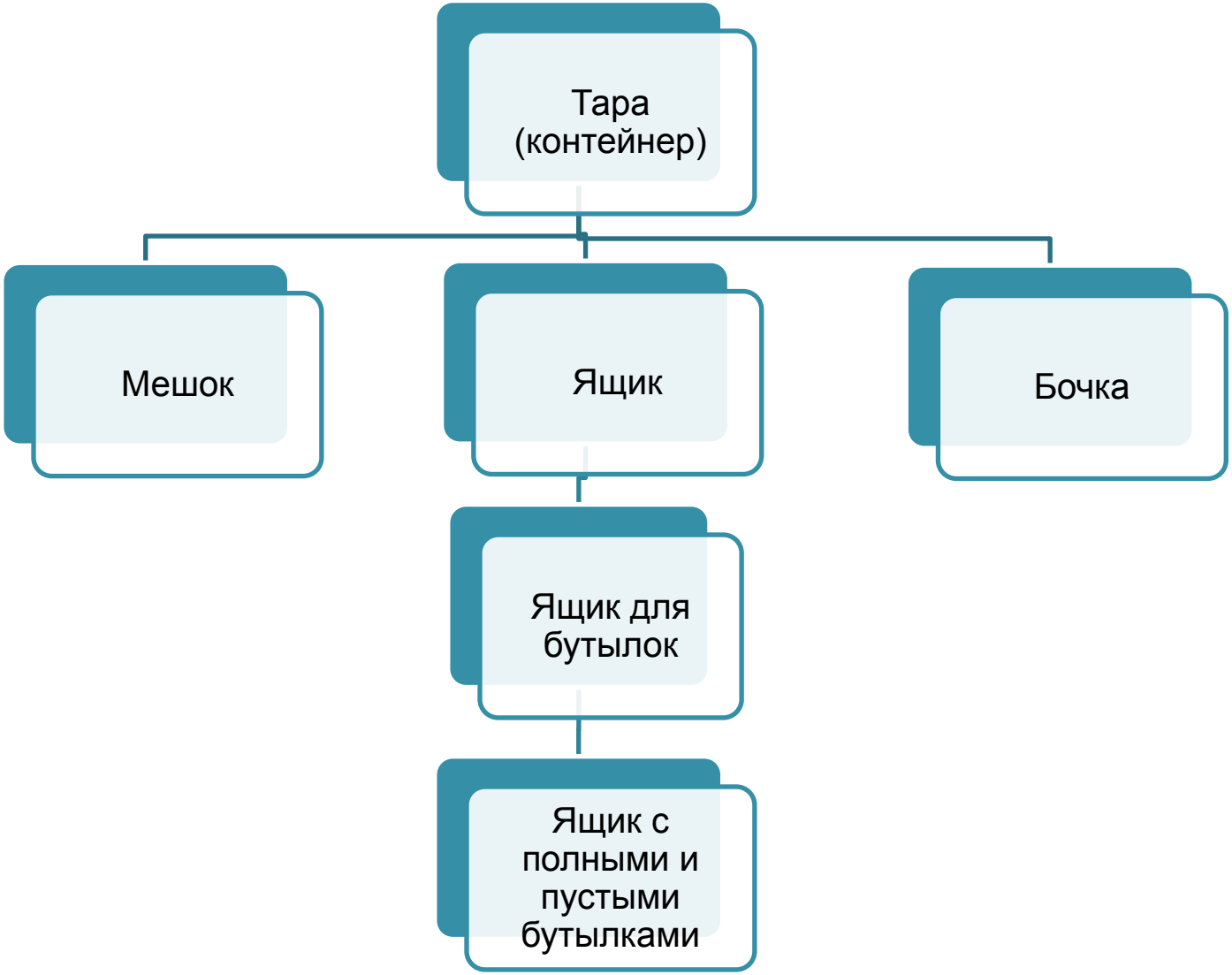
Что такое наследование?

Наследование – отношение между двумя классами (*предком* и *потомком*),когда объект-потомок повторяет элементы структуры и поведения предка.

Наследование призвано отображать иерархичность реального мира:



Пример иерархии понятий



Реализация наследования в ООП

В объектно-ориентированном программировании наследование реализуется с помощью возможности порождать один класс от другого, передавая порождаемому (производному) классу от класса-предка (базового класса) все поля и методы и добавляя, при необходимости, новые поля и методы.

В языке C++ для класса может существовать несколько непосредственных предков (родителей)

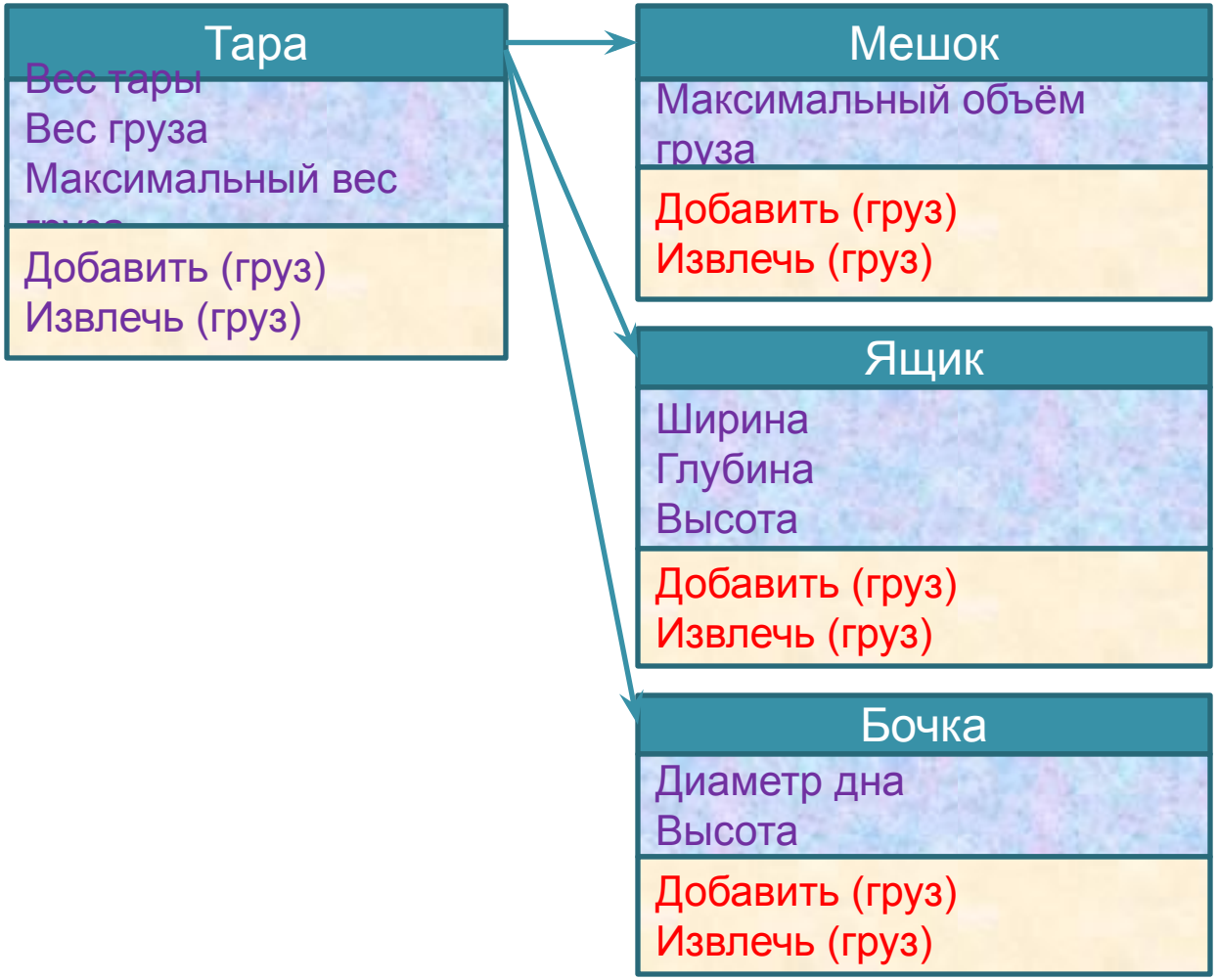
Возможности наследования

- Добавлять в производном классе данные, которые представляет базовый класс
- Дополнять в производном классе функциональные возможности базового класса
- Модифицировать в производном классе методы базового класса

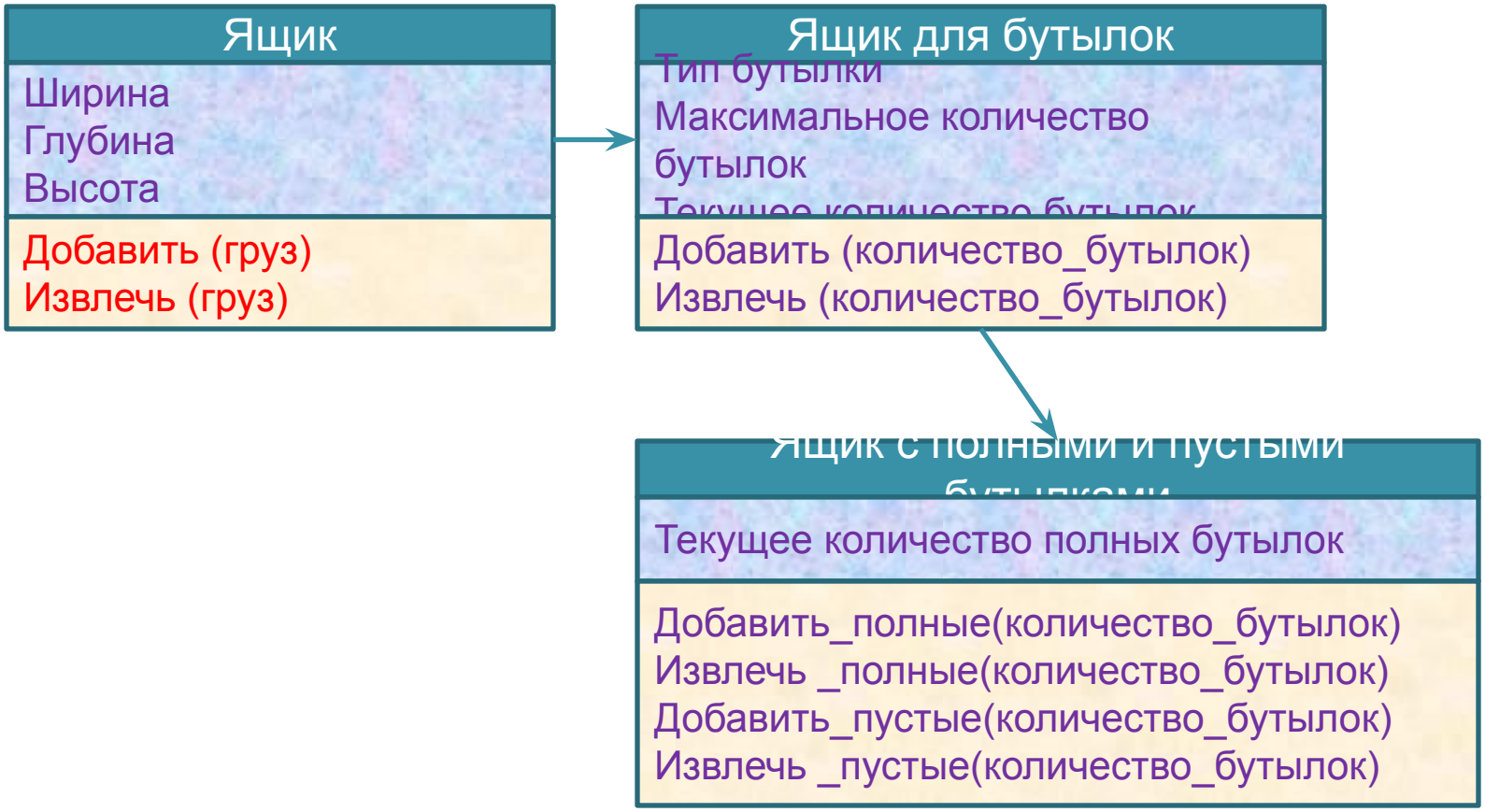
НЕЛЬЗЯ:

- Модифицировать в производном классе данные, представленные базовым классом (сохранив их идентификаторы)

Пример передачи полей и методов при наследовании



Пример передачи полей и методов при наследовании



Спецификатор доступа **protected**

Для того, чтобы организовать эффективную работу для программистов – авторов классов-потомков, введен специальный спецификатор доступа protected.

Полный список спецификаторов доступа:

- **private:** - члены класса, доступные только разработчикам класса (т.е. только при реализации методов этого класса)
- **public: -** члены класса, доступные как разработчикам, так и пользователям класса
- **protected:** - члены класса, доступные разработчикам класса и разработчикам классов-потомков

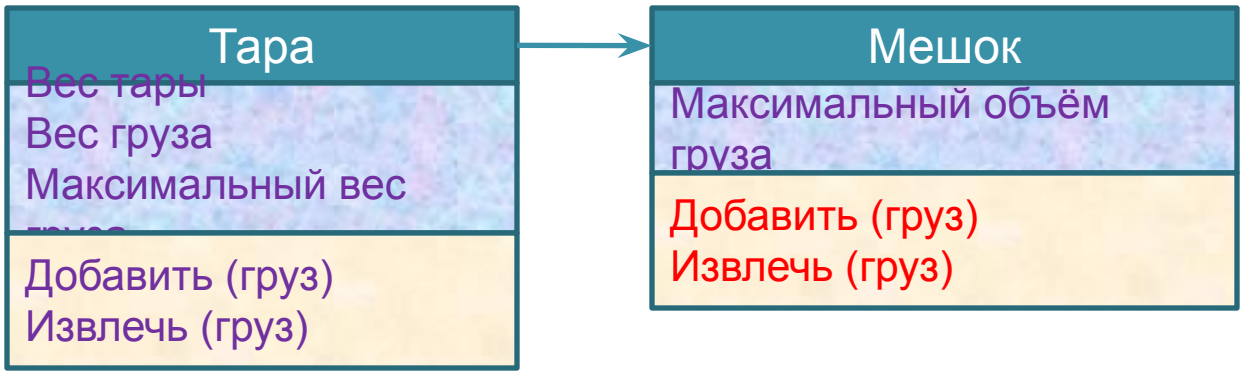
Реализация наследования

```
class имяклассанаследника :
private | public | protected // ключидоступа
базовыйкласс
{телокласса};
```

ключ доступа	спецификатор базового класса	доступ в классе-наследнике
private	private protected public	нет private private
protected	private protected public	нет protected protected
public	private protected public	нет protected public

Переопределение и перегрузка

- **Переопределение метода** – ситуация, когда в классе-потомке определяется метод с таким же названием и набором параметров, как и у предка. Для предка и потомка работают разные методы:



- **Перегрузка** – определение в одном и том же классе нескольких методов с различным набором параметров

Пример наследования: точка в трёхмерном пространстве

```
class Point3D: public Point2D{
private:
  double z;
public:
  Point3D(double=0, double=0, double=0);
  double GetZ() const;
  void SetZ(double az);
  double Module() const;
…
};
```

Реализация класса Point3D

```
Point3D::Point3D(double ax, double ay, double az):
  Point2D(ax, ay), z(az){}
double Point3D::GetZ() const { return z; }
void Point3D::SetZ(double az) {z = az;}
double Point3D:: Module() const
{ return sqrt(x*x + y*y + z*z); }
// не будет компилироваться из-за запрета
// на доступ к x и y, если эти поля не объявлены
// со спецификатором protected

// Правильный вариант:
double Point3D::Module() const {
  double x = GetX(), y = GetY();
  return sqrt(x*x + y*y + z*z);
)
```

Доступ к методам предка

Для доступа к методам предка, переопределённым в классе-потомке, используется операция ::

Пример: получить координату **ρ** для цилиндрической системы координат (**ρ**, **ϕ**, **z**)

```
double Point3D::GetRo () const {
  return Point2D::Module();
}
```

Особенности реализации наследования

- Конструкторы не наследуются;
- Деструктор не наследуется;
- Переопределённая операция присваивания не наследуется;
- Нет класса – прародителя всех классов.

Соответствие и преобразование типов при наследовании

- В левой части операции присваивания может стоять предок, а в правой – потомок (но не наоборот!)
- В левой части операции присваивания может стоять указатель на предка, а в правой – указатель на потомка (но не наоборот!)
- С помощью операции **new** можно указателю на предка присвоить адрес памяти, выделенной потомку (но не наоборот);
- Если формальный параметр функции – предок (указатель или ссылка), то фактическим параметром может быть потомок (указатель или ссылка).

Соответствие и преобразование типов при наследовании (примеры)

```
Point2D p(3,7);
Point2D* cp;
Point3D q(2, 9, 4);
Point3D* cq;

p = q; // можно
q = p; // нельзя

cp = cq; // можно
cq = cp; // нельзя

cp = new Point3D(); // можно
cq = new Point2D(); // нельзя
```

Соответствие и преобразование типов при наследовании (примеры)

```
void f1(Point2D x) {}
void f2(Point2D* x) {}
void f3(Point2D& x) {}

Point3D q(2, 9, 4);
Point3D* cq;

f1(q);   // можно во всех трёх случаях
f2(cq);
f3(q);
```