Наследование

Понятие наследования

Уровни доступа при наследовании

Использование наследования в QT библиотеке

Понятие наследования

Наследование – это такое отношение между классами, когда один класс повторяет структуру и поведение другого класса.

Другими словами, новый класс получает свойства и поведение уже существующего класса.

Наличие механизма наследования отличает объектно-ориентированные языки от объектных.

Назначение наследования

Наследование дает возможность заключить некоторое общее или схожее поведение различных объектов в одном базовом классе.

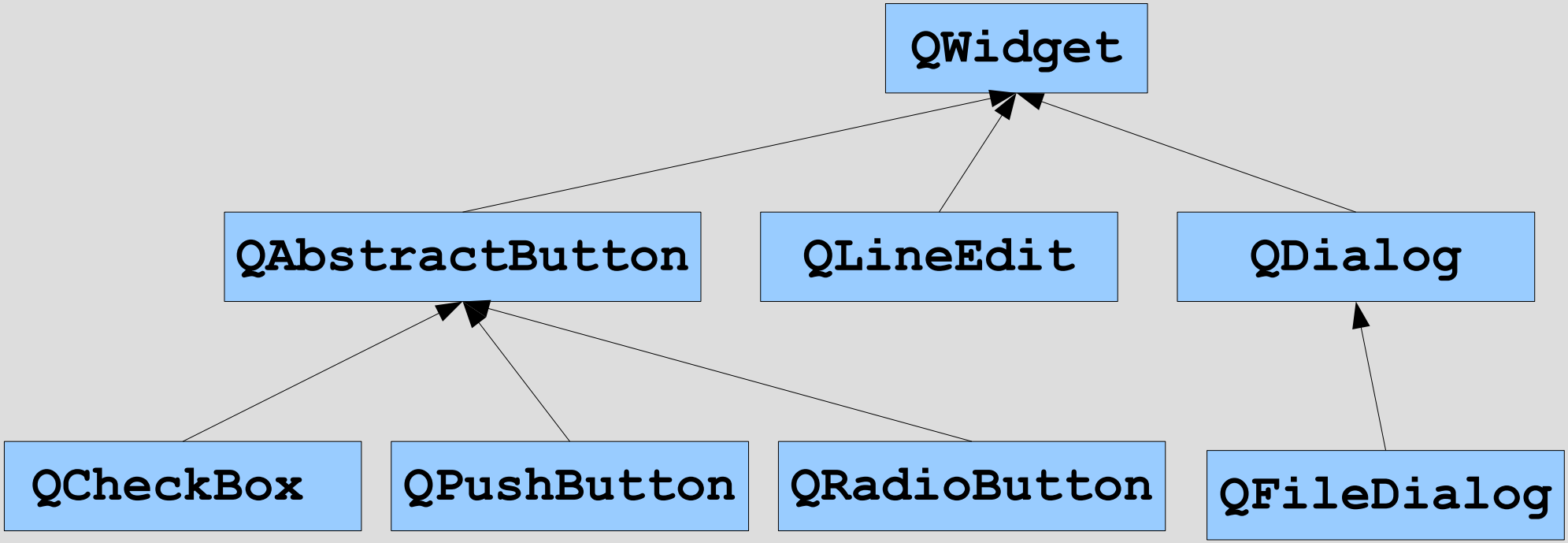
Несколько классов будут затем наследовать это поведение, являясь производными от базового.

Понятие родительского и дочернего классов

Класс, структура и поведение которого наследуется, называется суперклассом, надклассом, базовым или родительским классом.

Класс, производный от суперкласса, называется подклассом, производным или дочерним классом.

Примеры наследования



Формы наследования

В подклассе структура и поведение исходного суперкласса могут дополняться, переопределяться или ограничиваться.

Исходя из этого, можно выделить различные формы наследования.

Формы наследования

Специализация

Спецификация

Ограничение

Специализация

Дочерний класс является более конкретным, частным или специализированным случаем родительского класса.

Между дочерним и родительским классом возникает отношение «is-a» или «быть экземпляром» или «обобщение – специализация».

Поведение (методы) базового класса, в основном, переопределяется.

Примеры специализации

`QStringList` является специализированным случаем `QList`

`QLineEdit` **является специализированным случаем** `QWidget`

`QFileDialog` является специализированным случаем `QDialog`

Спецификация

Родительский класс описывает поведение, которое реализуется в дочернем классе, но оставлено нереализованным в родительском.

В таких случаях родительский класс называют абстрактно-специфицированным классом.

От абстрактного класса порождать объекты запрещено.

Пример спецификации

Класс `QAbstractButton` является абстрактным, т.к. в нем не реализован метод отрисовки:

```
void paintEvent(QPaintEvent * event)
```

Так как класс `QAbstractButton` представляет собой общее понятие кнопки, у которой нет «образа», то метод не имеет реализации.

Класс `QCheckBox` является специализированным случаем кнопки, известен его «образ», а потому в нем указанный метод реализован.

Задание

Даны классы:

- `MyCircle` - «окружность»
- `MyEllipse` - «эллипс»
- `MyOlimpicRings` - «олимпийские кольца»

Укажите, имеются ли между ними отношения наследования. Если оношения наследования имеются, то укажите их направление и форму.

«Лакмусовая бумажка» наследования – это обратная проверка: если B не есть A, то B не стоит производить от A.

Решение

Утверждение «окружность не есть частный случай эллипса» неверно, поэтому класс **`MyCircle`** должен наследоваться от **`MyEllipse`**.

Так как от класса **`MyEllipse`** можно порождать объекты и все методы класса определены, то между классами наблюдается отношение специализации.

Утверждение «олимпийские кольца не есть частный случай окружности» верно, поэтому класс **`MyOlimpicRings`** не должен наследоваться от **`MyCircle`**.

Синтаксис наследования при специализации и спецификации

Специализация и спецификация реализуются через открытое наследование.

```
class Base
{
    // Описание базового класса
};

// Открытое наследование
class Derived: public Base
{
    // Описание производного класса
};
```

Уровни доступа к элементам класса при открытом наследовании

Уровень доступа к элементу в базовом классе	Спецификатор доступа при наследовании	Уровень доступа к элементу в производном классе
private	public	не доступен
protected		protected
public		public

private – элементы класса доступны только внутри класса

protected – элементы класса доступны внутри базового класса и во всех производных от него классах

public – элементы классы доступны везде, в то числе, и вне класса

Задание

Имеется базовый класс **`MyEllipse`**. Создайте производный от него класс **`MyCircle`**. Считается, что класс **`MyCircle`** обладает всеми свойствами и методами класса **`MyEllipse`**.

Объявление наследования

```
class MyEllipse
{
    // Описание базового класса
};

class MyCircle: public MyEllipse
{
    // Описание производного класса
};
```

Ограничение

Дочерний класс ограничивает использование некоторых свойств и методов родительского класса.

Для реализации отношения «ограничение» используется закрытое и защищенное наследование.

Пример ограничения

`QDateEdit` является специализированным случаем `QDateTimeEdit`, однако в нем должны отсутствовать свойства и методы для работы со временем.

В действительности, классы `QDateEdit` и `QDateTimeEdit` ничем не отличаются, т.к. при наследовании не используется закрытое или защищенное наследование.

Обобщенный синтаксис наследования

```
class Base
{
    // Описание базового класса
};


class Derived: спецификатор_доступа Base
{
    // Описание производного класса
};
```

Спецификатор доступа при наследовании

Спецификатор доступа, который указывается при наследовании, определяет уровень доступа к элементам базового класса в производном классе, что иллюстрируется следующей схемой.

Открытое, защищенное и закрытое наследование

Уровень доступа к элементу в базовом классе	Спецификатор доступа при наследовании	Уровень доступа к элементу в производном классе
private	public	не доступен
protected		protected
public		public
private	protected	не доступен
protected		protected
public		protected
private	private	не доступен
protected		private
public		private

Задание

Используя библиотечный класс **QLinkedList<T>**, реализовать класс **IntStack**, который представляет собой стек целых чисел. В указанном классе реализовать следующие методы:

```
void push(int t);// втолкнуть знач. в стек
int pop ();      // вытолкнуть знач.из стека
int size();      // получить кол-во значений
```

Скрыть в классе **IntStack**, унаследованные методы от класса **QLinkedList<T>**, не подходящие для данного класса.

Справочная информация по классу QLinkedList

`void append ( const T & value )` - добавить значение в конец списка

`T & last ()` - возвращает последнее значение в списке

`void removeLast ()` - удалить последнее значение из списка

`int size () const` - возвращает кол-во значений в списке

Пример защищенного наследования

```
// Так как наследование закрытое, то все
// методы класса QLinkedList теперь не
// доступны из вне класса
class IntStack: protected QLinkedList <int>
{
public:       // объявляем новые
              // общедоступные методы
  // втолкнуть знач. в стек
  void push (int val);
  // вытолкнуть знач.из стека
  int pop ();
  // получить кол-во значений
  int size();
};
```

Пример защищенного наследования

```
// втолкнуть знач. в стек
void IntStack::push (int val)
{
  append(val);
}

// получить кол-во значений
int IntStack::size()
{
  // Внимание!!! Вызываем метод базового
  // класса, поэтому указываем префикс
  return QLinkedList<int>::size();
}
```

Пример защищенного наследования

```
// вытолкнуть знач.из стека
int IntStack::pop ()
{
   int val = 0;

   if(size() > 0)
   {
      val = last();
      removeLast();
   }

   return val;
}
```

Пример защищенного наследования

```
// вытолкнуть знач.из стека
int pop ()
{
   int val = 0;

   if(size() > 0)
   {
      val = last();
      removeLast();
   }

   return val;
}
```