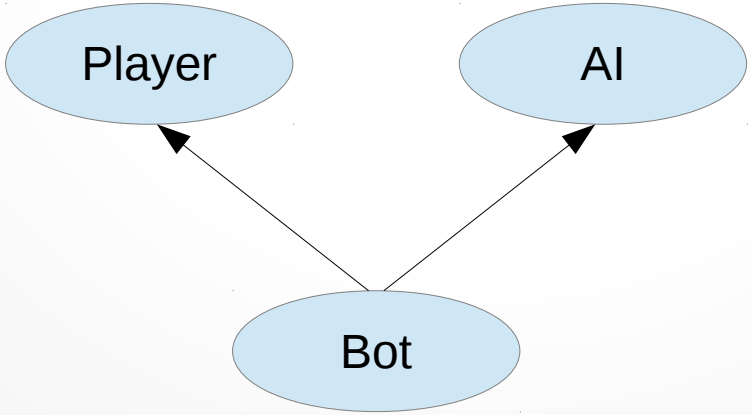
Множествено наследяване

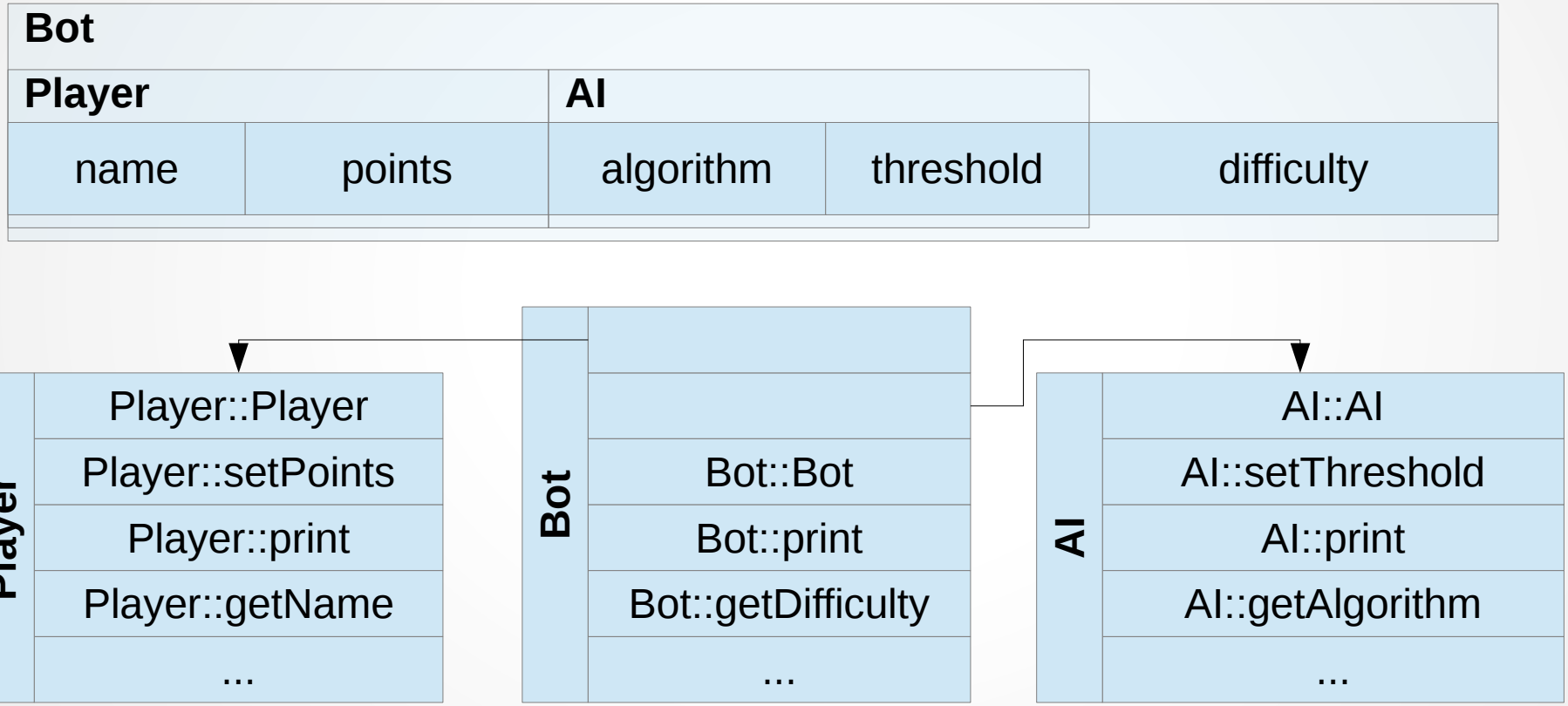
Част 1

Какво е множествено наследяване?

- Производен клас с повече от един основен клас
- Производният клас комбинира характеристиките и поведението на всичките си основни класове
- Пример:
  class Bot : public Player, public AI { ... };



Физическо представяне



Жизнен цикъл на обект от производен клас

- За обекта са заделя памет (на стека или в динамичната памет)
- Извиква се конструктор, който
  - извиква конструкторите на основните класове в реда на наследяване
  - извиква конструкторите на всички съдържани обекти
- ... (работа с обекта)
- Достига се края на областта на действие на обекта
- Извиква се деструктор, който
  - извиква деструкторите на всички съдържани обекти
  - извиква деструкторите на всички основни класове в ред обратен на реда на наследяване
- Паметта на обекта се освобождава

Конструктори

- Конструкторите на производния клас трябва да указват как се конструират всяка една от наследените части
  - ако за някой от основните класове не е указан кой конструктор да се извика, тогава се извика този по подразбиране
- Пример:

```
Bot::Bot(char const* _name, int _pts, char const* _algo,
                            double _threshold, int _difficulty)
 : Player(_name, _pts), AI(_algo, _threshold), difficulty(_difficulty)
```

Деструктори

- Деструкторите на основните класове се викат автоматично

Предефинирани функции

- Предефинираните функции могат да извикат съответна функция във всеки един от основните класове
- Пример:

```
void Bot::print() const {
  Player::print();
  AI::print();
  cout << „Ниво на трудност: „ << difficulty << endl;
}
```

Голямата четворка

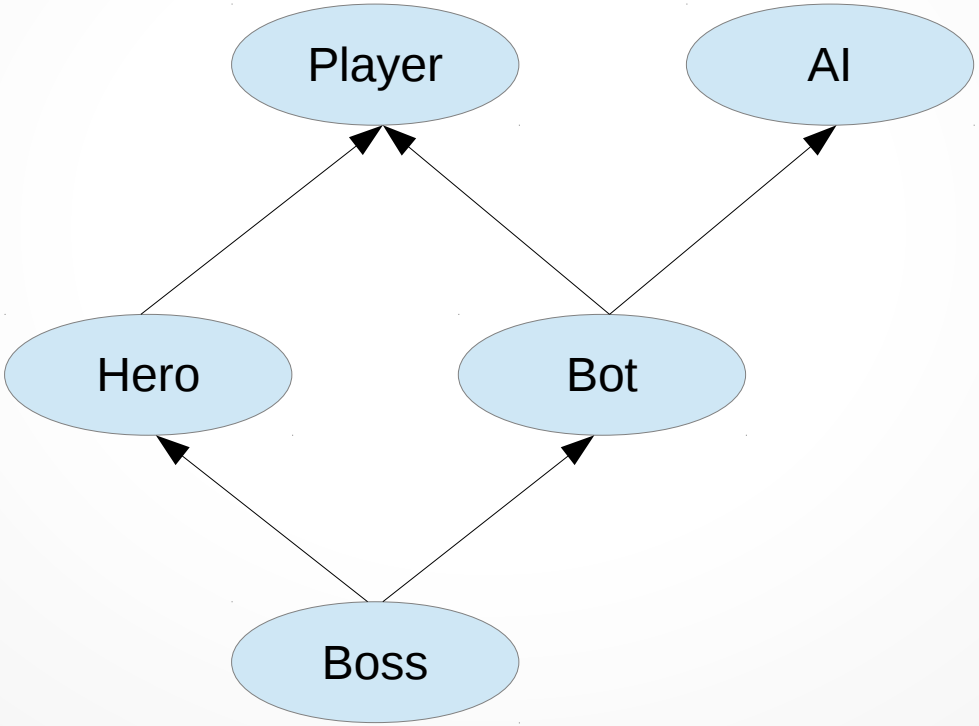
- Системно генерираните методи от голямата четворка правят това, което трябва
- Конструкторът по подразбиране на производния клас извиква съответните конструктори по подразбиране на основните класове
- Конструкторът за копиране на производния клас извиква съответните конструктори за копиране на основните класове
- Операцията за присвояване на производния клас извиква съответните операции за присвояване на основните класове
- Винаги редът е същият като реда на наследяване

Проблеми при множественото наследяване

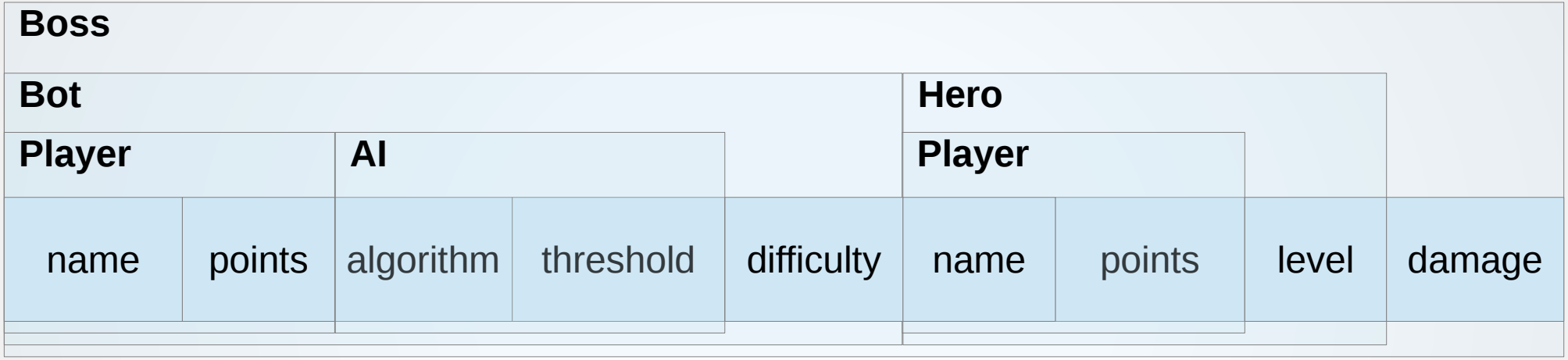
- Усложнява се йерархията (става по-трудна за поддържане)
- Двусмислици при обръщения към компоненти на клас
  - коя от няколкото наследени компоненти се има предвид?
- Пример:
  bool Player::operator>(Player const& p) const
  { return points > p.points; }
  bool AI::operator>(AI const& ai) const
  { return threshold > ai.threshold; }
- Bot bot1(„Deep Thought“, 20, „minimax“, 3.8, 5),
  bot2(„HAL 9000“, 40, „alpha-beta“, 1.9, 15);
- if (bot1 > bot2) cout << bot1 << „ е по-добър“;

Косвено повторно наследяване

- Deadly Diamond of Death



Физическо представяне



- Всяка компонента на Player се повтаря
- Коя наследена компонента ще върне (Player)boss?
- Ако Boss b, какво ще върне b.getName()?
- Раздвояване на личността!
- Какво всъщност искаме да се получи?

Желаното физическо представяне

Boss						
Player		**Hero**	**Bot**			
			AI			
name	points	level	algorithm	threshold	difficulty	damage

- Искаме да разрешим нееднозначността като поддържаме единствено копие на общия пра-основен клас
- Проблем: нарушаваме структурата на йерархията!
- Ако преобразуваме Bot до Bot, няма да можем да го преобразуваме после до Player!