Тема 8.

Статични член-данни. Изключения. Обработка на изключения. Йерархия на изключенията и примери. Изключения в конструктори и деструктори. Нива на exception safety.
Пример с клас, който брои инстанциите си.

Static

- **статична функция** → може да бъде използвана от други .cpp файлове чрез forward declaration
  - → не можем да дефинираме друга ф-я със същото име дори в друг файл
  - → ограничава се до 1 комп. единица (.obj)
  - → еквивалент на функция в анонимен namespace

- **статична променлива в тялото на функция**
  - → създава се при първото за обекта извикване на функцията
  - → общa е за ∀ извиквания на функцията
  - → изтрива се при излизане от програмата

- **статични член-променливи на клас**
  - → глобален обект, енкапсулиран в клас
  - → не е обвързан с конкретна инстанция
  - → създава се и се изтрива веднъж
  - → не влияе на големината на обекта

→ има достъп до член-данните
→ класът играе ролята на интерфейс

- статична член-функция
  → външна функция, не ни трябва обект, за да я извикаме
  → има достъп само до статичните член-данни на класа
  → не може да е const / virtual

- статичен клас - клас, който има само статични член-данни

Изключения (exceptions) - сигнал, че има проблеми, начин за комуникация между функции

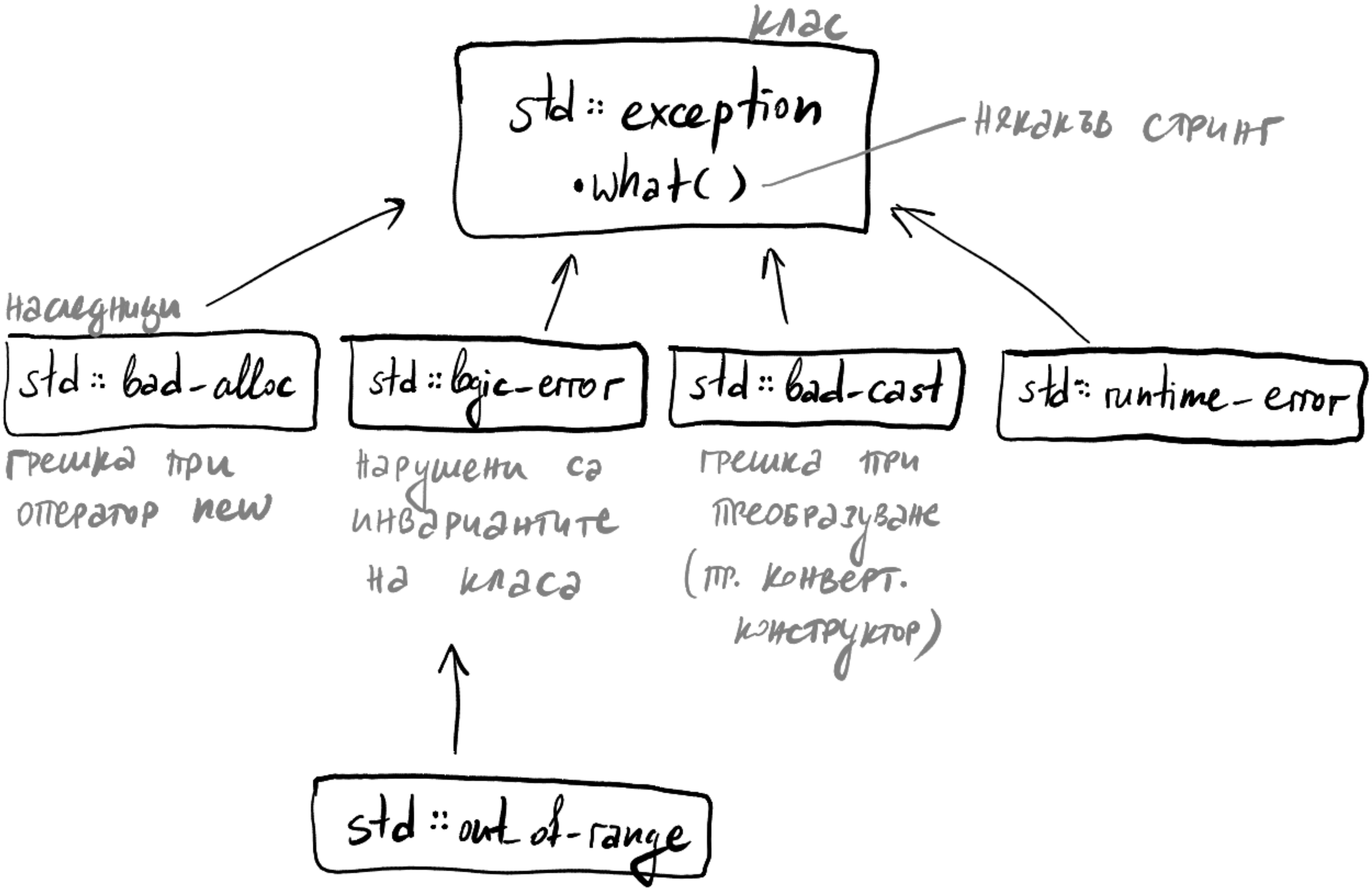
→ throw <обект>
→ throw спира функцията, започва stack unwinding
(ф-я трябва отговорността за грешката на тази над нея, ако никой не я обработи => std::terminate)

→ обработка чрез try-catch блок

```
try {
   // проблемен код
}
catch (int x) {
   // обработка
}
```

На мястото на int може да седи всякакъв променлив или (...), което хваща всеки тип

→ Йерархия на изключенията



→ Изключения в конструктори

Ако хвърлим грешка в конструктора на обект A, не се извиква деструкторът, защото A не е напълно създаден. Това може да доведе до утечка на памет. При хвърляне на изключение в конструктора се извикват деструкторите на ∀ напълно построени обекти в класа

→ Изключения в деструктора

```
class A {
    ~A() { throw 37; }
};
```

```
int g() {
  A obj;
  throw 5;
}
```

Не хвърляме изкл. в деструктора, защото може да се получат 2 или повече активни изключения
⇒ std :: terminate

→ Нива на сигурност при изключения

1) no throw guarantee - класът / функцията не хвърля изключение в никаков случай

(move констр., size(), empty())

2) strong exception safety - може да се хвърли грешка, но няма да повлияе на състоянието на обекта

(push_back - при добавяне на 11 елемента, ако 11-тото се провали няма да повлияе на предишните 10)

3) basic exception safety - възможно е да хвърли изключение, но ще остане във валидно състояние

4) no exception guarantee - никакви обещания