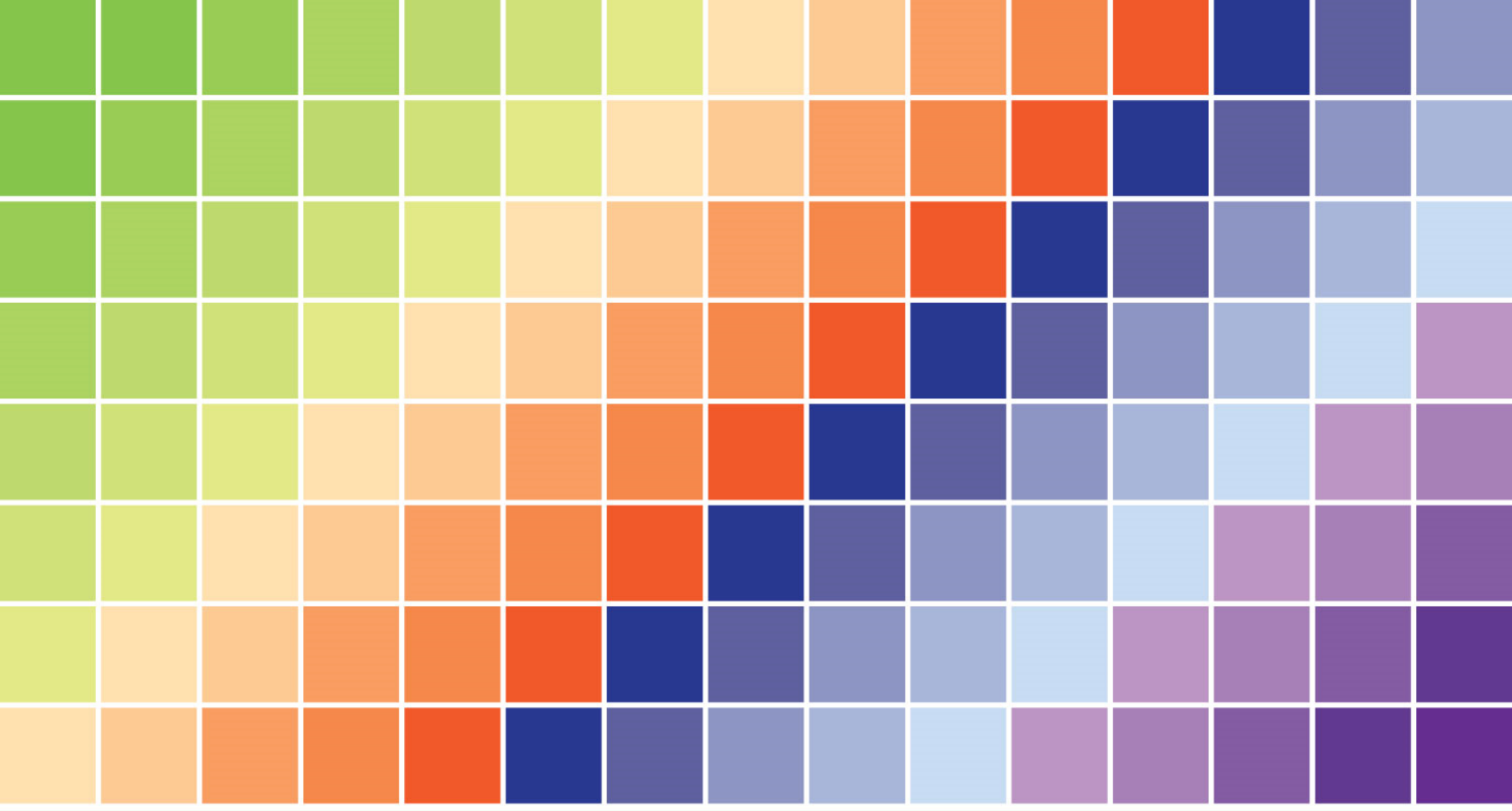


School of Computer Science

# ОБСУЖДЕНИЕ ПРЕДЫДУЩЕГО ЗАНЯТИЯ

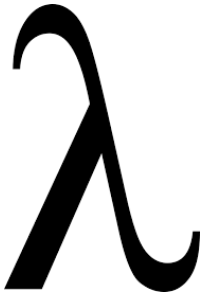
# ФУНКЦИОНАЛЬНАЯ ПАРАДИГМА
## ПРЕДПОСЫЛКИ

- В 30-ых годах 20-го века в Принстоне собрались Алан Тьюринг, Джон фон Нейман, Курт Гёдель и **Аллонзо Чёрч**

- Интересуясь формальными системами вычислений они пытались ответить на такие вопросы:
  - Какие задачи можно решать на машине с бесконечными вычислительными возможностями?
  - Можно ли решать эти задачи автоматически?
  - Существуют ли неразрешимые задачи и почему?
  - Будут ли машины с разной архитектурой одинаковыми по мощности?

# ФУНКЦИОНАЛЬНАЯ ПАРАДИГМА
## ТЕОРИЯ

- Использует математическую теорию лямбда-исчисления Алонзо Чёрча (1936)
- Основана на функциях, которые принимают в качестве аргументов функции, и возвращают функцию
- Такая функция была обозначена греческой буквой Лямбда, что дало название всей системе



# ФУНКЦИОНАЛЬНАЯ ПАРАДИГМА
## ПРАКТИЧЕСКОЕ ВОПЛОЩЕНИЕ

- В конце 50-ых Джон Маккарти стал проявлять интерес к работам Черча

- В 1958 году он представил язык обработки списков **Lis**t **P**rocessing Language (Lisp)

- Lisp – имплементация Лямбда-исчисления Алонзо, которая работает на машинах с архитектурой фон Неймана
- 1973 г построена аппаратная Lisp-машина

# ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

## ПРИМЕР ПРОГРАММЫ НА LISP

```
(defun encr(l,a)
   (cond
         ((null l) l
         )
         ((atom (car l))
           (cond
             ( (> (car l) 0) (cons (+ (car l) a)  (encr (cdr l) a) )
             )
             (t (encr (cdr l) a) )
           )
         )
         ( t
            ( cons (encr (car l) a) (encr (cdr l) a))
         )
  )
)

(defun main()
  (setq l '(1 -2 (3 5) -2 4 (6 -2)))
  (setq l1 (encr l 1))
)
(main)
```

# ФУНКЦИОНАЛЬНЫЙ ПОДХОД

- Программа (её фрагмент) рассматривается как вычисление математических функций
- Не используются состояния (переменные)
- Функции являются «чистыми», т.е. они:
  - Не меняют внешние переменные
  - Ничего никуда не посылают, не принимают извне, не сохраняют и не печатают
  - Делают вычисления, учитывая только аргументы, и возвращают новые данные
  - При этом отсутствуют какие-либо побочные эффекты, только возвращается результат
- ФП в Java включает:
  - **Лямбда-функции**
  - **Stream API**

# ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

## ЗАЧЕМ?

- Возможность написания программ, работающих параллельно:
  - Конкурентность/параллелизм/многопоточность – одновременное выполнение нескольких вычислительных процессов, которые взаимодействуют друг с другом
- «Чистота» функций (нет побочных эффектов), позволяет кэшировать их результаты для последующего использования
- Используется в задачах с высокой вычислительной сложностью
- **Функциональный подход упрощает работу с коллекциями в Java**

# ОПРЕДЕЛЕНИЯ
## ЛЯМБДА-ФУНКЦИЯ В JAVA

- **Лямбда функция (выражение)** – это анонимный блок кода с параметрами
- Такие функции удобно использовать везде, где требуется объект класса, реализующего интерфейс ровно с одним методом

```
(параметры) -> {
   блок команд
}
или
(параметры) -> одно_выражение
```

# ВОПРОС ПО ЛЯМБДА-ФУНКЦИИ

Что из перечисленного относится к характеристикам лямбда-функции в Java?

1. Не имеет имени
2. Обязательно имеет возвращаемое значение
3. Всегда состоит из одной строки
4. Может использоваться везде, где требуется объект класса, реализующего интерфейс с одним методом
5. Может использоваться как аргумент функции, только если тип аргумента указан как «лямбда»

# ВОПРОСЫ ПО ПРОШЛОМУ ЗАНЯТИЮ

Что такое исключение в программировании:

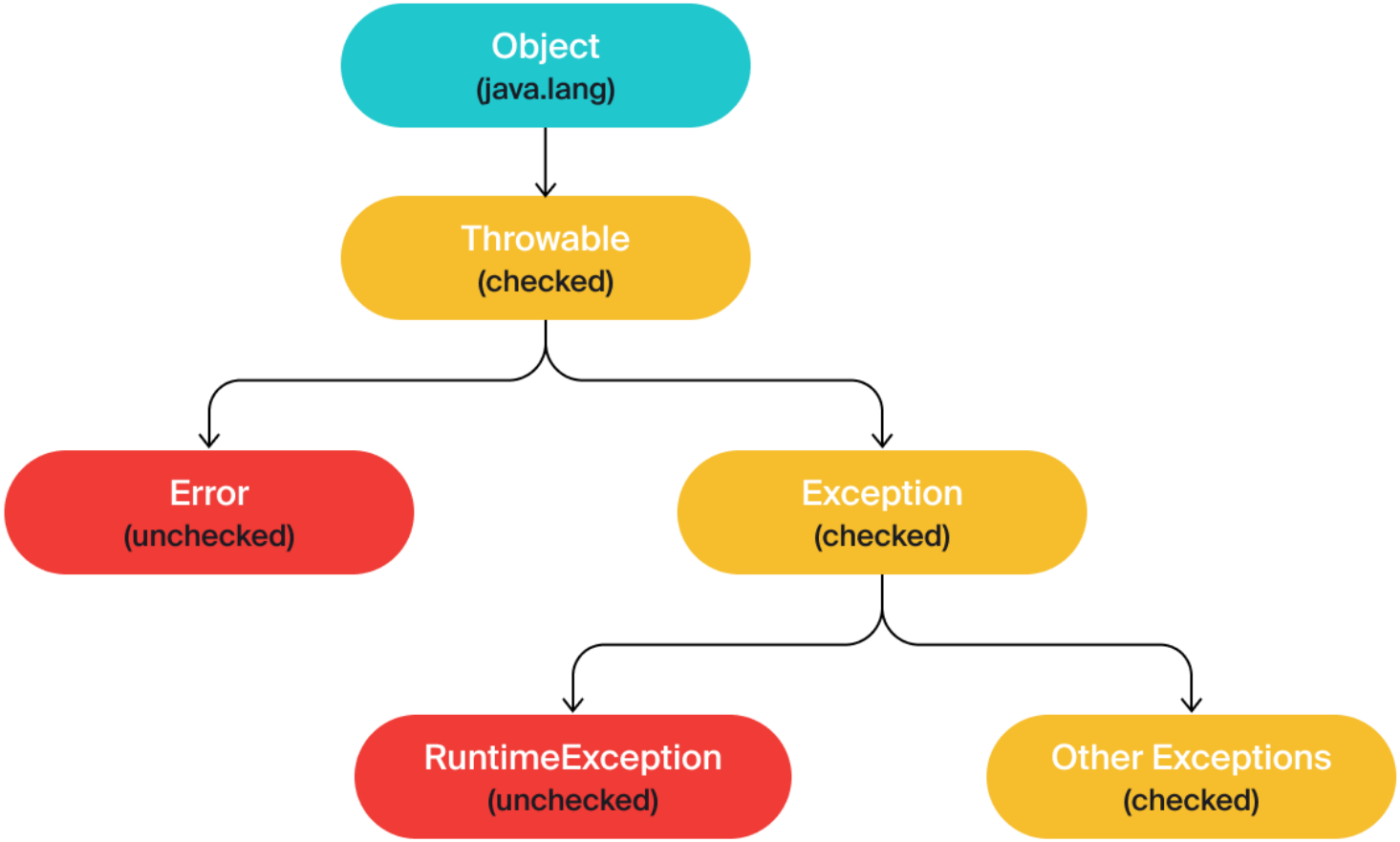
1. Исключительно хороший код, которому можно только позавидовать
2. Механизм обработки ошибочных ситуаций
3. Ошибка, при возникновении которой программа перестает работать

# ВОПРОСЫ ПО ПРОШЛОМУ ЗАНЯТИЮ

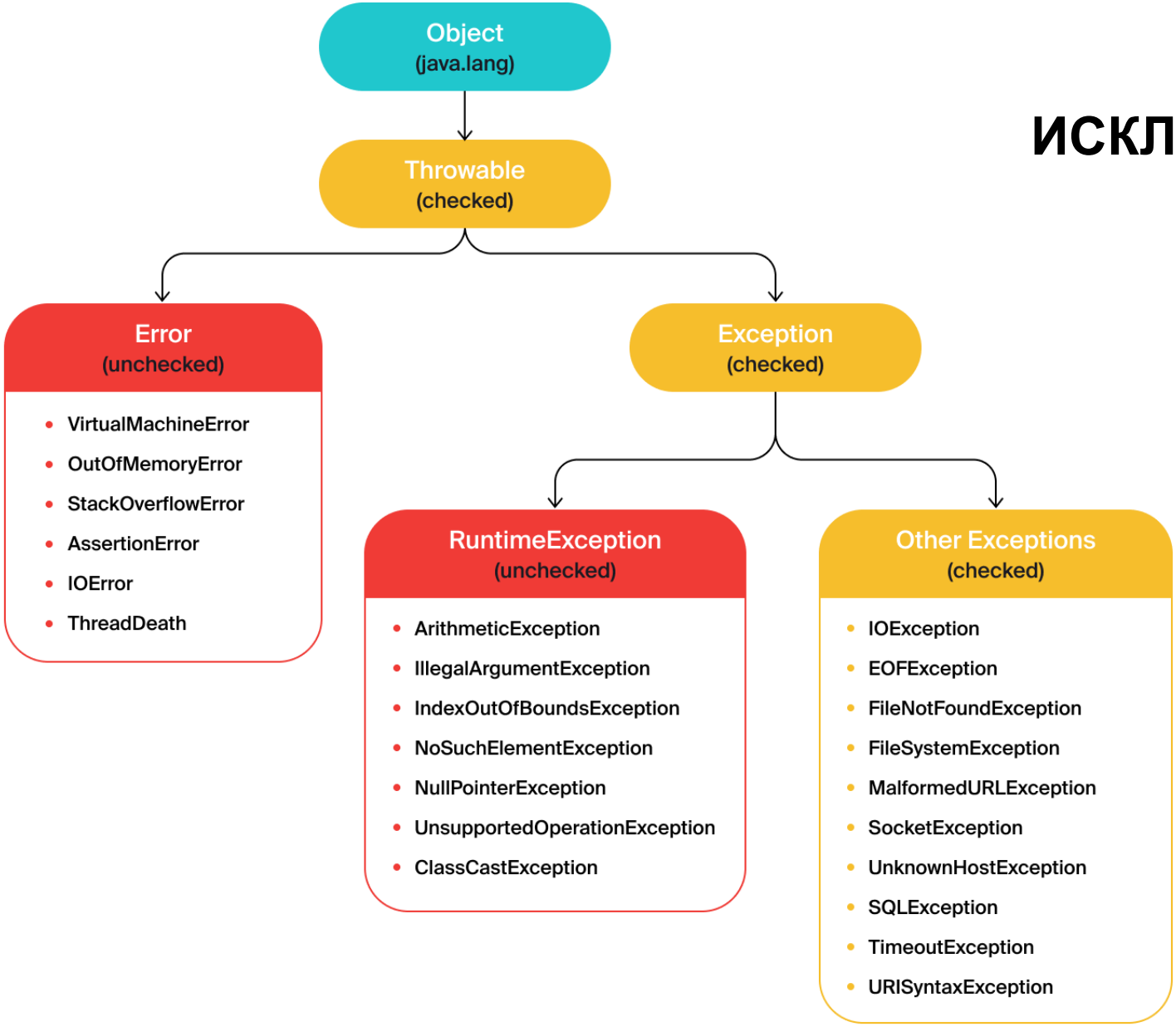
– Какие из этих слов применяются при обработке исключений в Java?

1) disaster
2) try
3) problem
4) catch
5) except
6) finally

# ИЕРАРХИЯ ИСКЛЮЧЕНИЙ В JAVA



# ИЕРАРХИЯ ИСКЛЮЧЕНИЙ В JAVA



# ВОПРОС ПО ИСКЛЮЧЕНИЯМ

Выберите верные утверждения:

1. Деление на ноль — это исключение непроверяемого типа
2. Проверяемые исключения — те, которые можно обработать, а непроверяемые — нет
3. Критические ошибки, которые приводят к серьёзным сбоям, как правило, относятся к непроверяемым исключениям
4. В логике программы должна быть обязательно предусмотрена обработка исключения NullPointerException
5. Exception генерирует непроверяемые исключения
6. Код, где есть необработанное проверяемое исключение, не скомпилируется

# СПАСИБО ЗА ВНИМАНИЕ !

# ВОПРОСЫ ?

School of Computer Science