VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Advanced Programming (CO2039)

Report (Semester 202, Duration: 01 weeks)

OOP vs FP

Advisor:	Mr. Lê Lam Sơn
Student Name:	Nguyễn Hoàng
Student ID:	1952255

HO CHI MINH CITY, AUGUST 2021

Contents

1 OOP and FP in baking a pizza

OOP makes code understandable by encapsulating moving parts. FP makes code understandable by minimizing moving parts.

What? Alright that sounds a bit rough, let's rephrase this a bit. OOP aims to model the world in self-contained entities, and affects change by modifying the state of itself or other entities. FP on the other hand aims to not modify the original data, but rather creates new data given some existing data.

To demonstrate this, we will try to make a pizza. With OOP, a big box or object with all the materials to create a pizza is available, and the helper methods will slowly transform them into a complete pizza. FP will take a different approach, as materials are given to each stage/step/activity in order to be used in the next activity until the final product is achieved.

We will try to describe this pizza making progress programmatically using C++ and Haskell.

Let's start with a complete C++ program

```cpp
#include <iostream>

class Pastry
{
public:
  // Skeleton of the process
  void bake_me_baby()
  {
    prepare_dough();
    preheat_oven();
    add_sauce();
    add_toppings();
    bake();
  }

protected:
  // Subclasses have to implement these methods
  virtual void prepare_dough() = 0;
  virtual void add_sauce() = 0;
  virtual void add_toppings() = 0;
  virtual void bake() = 0;

  // Subclasses can override the methods or leave them be
  virtual void preheat_oven()
  {
    std::cout << "Preheating the oven" << std::endl;
  }
  virtual void cut()
  {
    std::cout << "Cutting" << std::endl;
  }
};

```

```cpp
class Pizza : public Pastry
{
protected:
  /**
   * 0: raw
   * 1: dough prepared
   * 2: sauce added
   * 3: toppings added
   * 4: baked
   **/
  int state = 0;

protected:
  void prepare_dough()
  {
    std::cout << "Preparing the dough" << std::endl;
    state = 1;
  }
  void add_sauce()
  {
    std::cout << "Adding the sauce" << std::endl;
    state = 2;
  }
  void add_toppings()
  {
    std::cout << "Adding toppings" << std::endl;
    state = 3;
  }
  void bake()
  {
    std::cout << "Baking" << std::endl;
    state = 4;
  }
};

int main(int argc, char **argv)
{
  Pizza *pizza = new Pizza();
  pizza->bake_me_baby();
  delete pizza;
  return 0;
}
```

Output of this program

```
Preparing the dough
Preheating the oven
Adding the sauce
Adding toppings
Baking
```

One more time, but with Haskell

```
  print "Hello, World!"
```

Output of this program

```
Hello, World!
```