Design Patterns

Creational Patterns
Singleton
Fluent Builder

Singleton

```
class PrintSpooler {
public:
  static Company* getInstance();
  void print(Document* doc);
private:
  PrintSpooler();
}
----------------------------------------
static PrintSpooler* s_instance;
PrintSpooler* PrintSpooler::getInstance() {
  if (!s_instance) {
    s_instance = new PrintSpooler();
  }
  return s_instance;
}
```

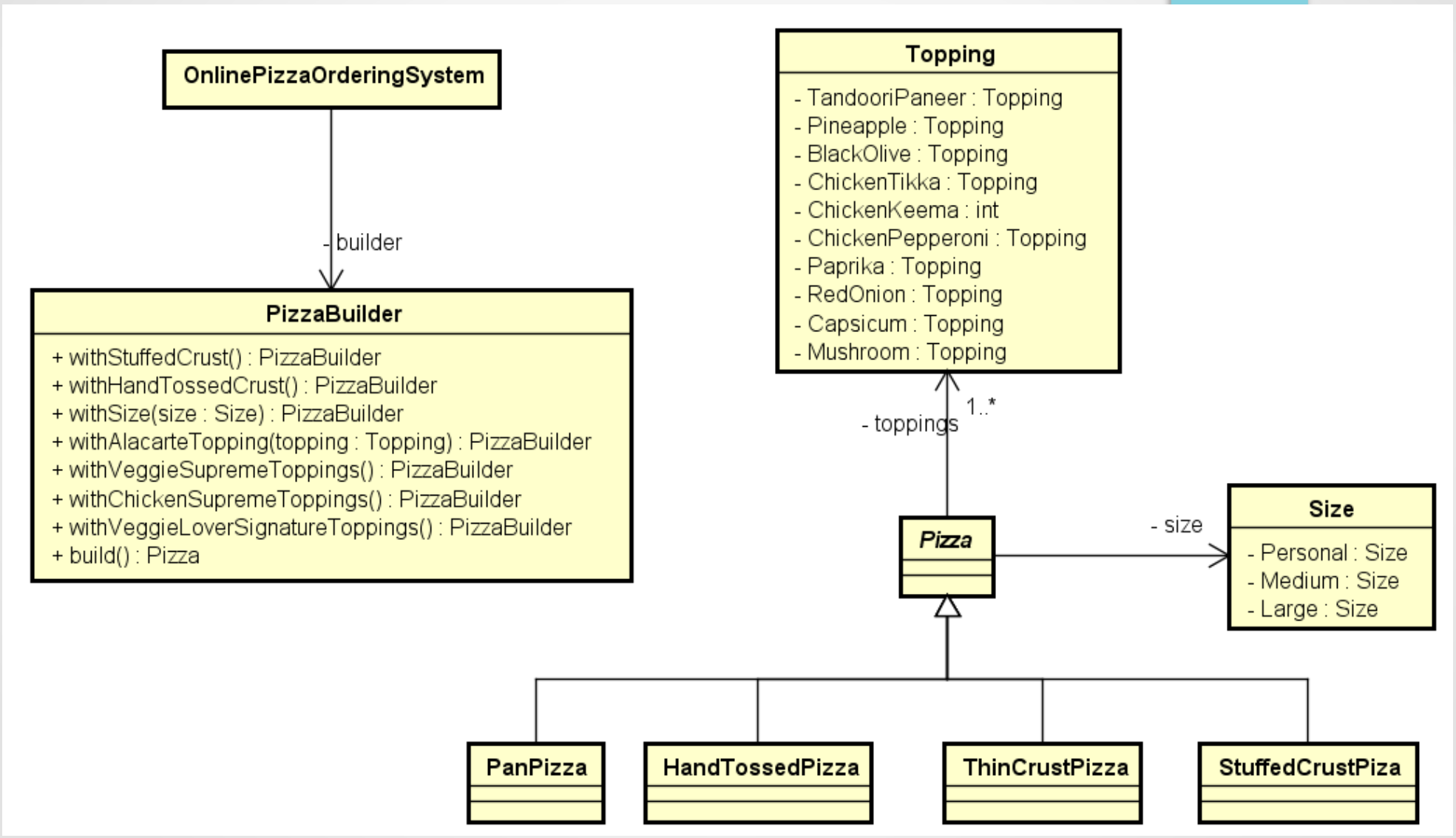
Singleton - A client view

```
int main() {

   Document * doc = new Document(..);
   ..
   ..
   PrintSpooler spool = PrintSpooler::getInstance();
   spool.print(doc);

}
```

Fluent Builder - A client view

```
class OnlinePizzaOrderingSystem {

  Pizza buildMyPizza() {

    PizzaBuilder builder =
                new PizaBuilder()
                ->withStuffedCrust()
                ->withSize(Size.Medium)
                ->withVeggieLoverSignatureToppings()

    Pizza piza = builder.build();

    return pizza;
  }

}
```

Fluent Builder



Fluent Builder Internals

```
class PizzaBuilder {

public:
  PrintSpooler();
  PizzaBuilder* withStuffedCrust();
  PizzaBuilder* withVeggieLoversSignatureToppings();
  Pizza* build();

}

----------------------------------------

PizzaBuilder* PizzaBuilder::withVeggieLoversSignatureToppings() {
  ...
  return this;
}

Pizza* PizzaBuilder::build() {
  if () {   ..     ..   return new StuffeCrustPizza(...);}

  else if () {   ..     ..   return new HandTossedPizza(...);}

}
```