Плюсы:

1) Использование объектно-ориентированного стиля.
2) Наименование переменных и методов выдержаны в одном стиле

```
// вещь
class Item : public QObject
{
    Q_OBJECT
    public:
        explicit Item(QObject *parent = 0);
        explicit Item(QString _id, QObject *parent = 0);
        virtual void Use();  // использовать
        QString getID();
        void prepare();  // подготовить к новому исп-ю
        bool isDisposeRequired();  // полностью ли исп-но
    protected:
        float status;  // состояние
        QString id;  // ID/Название
        bool clean;  // чистота
    signals:
        void isUsed();
        void isFinal();
    public slots:
};

class Plate : public Item
{
        Q_OBJECT
    public:
        explicit Plate(QObject *parent = 0);
        virtual void Use();
```

3) Использование стандартные структуры данных.

```
    QList<QPoint> findPath(QPoint src, QPoint dest);    // найти путь от точки до точки
    QList<QPoint> findFreeNeighbors(QPoint src);    // поиск свободных соседей
    QPoint findEntryPoint(QPoint src, QPoint dest); // поиск точки, к которой можно подойти
```

4) Использование шаблонов

```
template <typename T> void fillList(QList<T> *container, T src, int size)
{
    container->clear();
    for (int i = 0; i < size; i++)
        container->push_back(src);
}
```

Минусы:

1. Закомментированные участки кода

```
/*
void Model::cookDishes()
{
    needToCookMore = true;
}
*/
```

2. Большой блок try-catch, который ловит не понятно что, и не понятно почему возвращает false

```
                outp << c->getEntry(true);
            }
        }
        outp << employees.size();
        for (int i = 0; i < employees.size(); i++)
        {
            outp << employees.at(i)->getCoord();
            outp << dynamic_cast<Employee *>(employees.at(i))->getType();
        }
        outp << dishesAvailable.size();
        for (int i = 0; i < dishesAvailable.size(); i++)
        {
            Dish *d = dishesAvailable.at(i);
            outp << d->getName();
            outp << d->getPrice();
            outp << d->getQuantity();
            outp << d->getPopularity();
            for (int i = 0; i < d->getReqUtensils().size(); i++)
                outp << d->getReqUtensils().at(i);
        }
        outp << entrance;
    }
    catch(...)
    {
        return false;
    }
```

3. Низкая комментируемость. Те комментарии, что имеются, не дают представления о том, что происходит в коде.
4. Именование констант вопреки общепринятым соглашениям.

```
#define color_cashier QColor(255, 242, 0)
#define color_cashier_cap QColor(34, 177, 76)
#define color_cook QColor(195, 195, 195)
#define color_cook_cap QColor(255, 255, 255)
#define color_dishwasher QColor(0, 162, 232)
#define color_dishwasher_cap QColor(255, 255, 255)
```

5. Блоки case очень большого размера. Из-за этого трудно отследить логику

Оценка 5/10