1

MainWindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QVector>

#include "Task.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    void updateStatus();

public slots:
    void addTask();
    void removeTask(Task* task);
    void taskStatusChanged(Task* task);

private:
    Ui::MainWindow *ui;
    QVector<Task*> mTasks;
};

#endif // MAINWINDOW_H
```

MainWindow.cpp

```
#include "MainWindow.h"
#include "ui_MainWindow.h"

#include <QDebug>
#include <QInputDialog>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow),
    mTasks()
{
    ui->setupUi(this);
    connect(ui->addTaskButton, &QPushButton::clicked, this,
&MainWindow::addTask);
    updateStatus();
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

2

```
void MainWindow::addTask()
{
    bool ok;
    QString name = QInputDialog::getText(this, tr("Add task"),
                                         tr("Task name"), QLineEdit::Normal,
                                         tr("Untitled task"), &ok);
    if (ok && !name.isEmpty()) {
        qDebug() << "Adding new task";
        Task* task = new Task(name);
        connect(task, &Task::removed, this, &MainWindow::removeTask);
        connect(task, &Task::statusChanged, this,
&MainWindow::taskStatusChanged);
        mTasks.append(task);
        ui->tasksLayout->addWidget(task);
        updateStatus();
    }
}

void MainWindow::removeTask(Task* task)
{
    mTasks.removeOne(task);
    ui->tasksLayout->removeWidget(task);
    delete task;
    updateStatus();
}

void MainWindow::taskStatusChanged(Task* /*task*/)
{
    updateStatus();
}

void MainWindow::updateStatus()
{
    int completedCount = 0;
    for(auto t : mTasks)  {
        if (t->isCompleted()) {
            completedCount++;
        }
    }
    int todoCount = mTasks.size() - completedCount;

    ui->statusLabel->setText(QString("Status: %1 todo / %2 completed")
                             .arg(todoCount)
                             .arg(completedCount));
}
```

Task.h

```
#ifndef TASK_H
#define TASK_H

#include <QWidget>
#include <QString>

namespace Ui {
class Task;
}

class Task : public QWidget
{
```

```
    Q_OBJECT

public:
    explicit Task(const QString& name, QWidget *parent = 0);
    ~Task();

    void setName(const QString& name);
    QString name() const;
    bool isCompleted() const;

public slots:
    void rename();

signals:
    void removed(Task* task);
    void statusChanged(Task* task);

private slots:
    void checked(bool checked);

private:
    Ui::Task *ui;
};

#endif // TASK_H
```

Task.cpp

```
#include "Task.h"
#include "ui_Task.h"

#include <QInputDialog>
#include <QDebug>

Task::Task(const QString& name, QWidget *parent) :
        QWidget(parent),
        ui(new Ui::Task)
{
    ui->setupUi(this);
    setName(name);

    connect(ui->editButton, &QPushButton::clicked, this, &Task::rename);
    connect(ui->removeButton, &QPushButton::clicked, [this] {
        emit removed(this);
    });
    connect(ui->checkbox, &QCheckBox::toggled, this, &Task::checked);
}

Task::~Task()
{
    qDebug() << "~Task() called";
    delete ui;
}

void Task::setName(const QString& name)
{
    ui->checkbox->setText(name);
}

QString Task::name() const
```

```
{
    return ui->checkbox->text();
}

bool Task::isCompleted() const
{
   return ui->checkbox->isChecked();
}

void Task::rename()
{
    bool ok;
    QString value = QInputDialog::getText(this, tr("Edit task"),
                                          tr("Task name"), QLineEdit::Normal,
                                          this->name(), &ok);
    if (ok && !value.isEmpty()) {
        setName(value);
    }
}

void Task::checked(bool checked)
{
    QFont font(ui->checkbox->font());
    font.setStrikeOut(checked);
    ui->checkbox->setFont(font);

    emit statusChanged(this);
}
```

main.cpp

```
#include "MainWindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```