第一章 开发工具Intellij IDEA

IDEA是一个专门针对Java的集成开发工具（IDE），由Java语言编写。所以，需要有JRE运行环境并配置好环境变量。

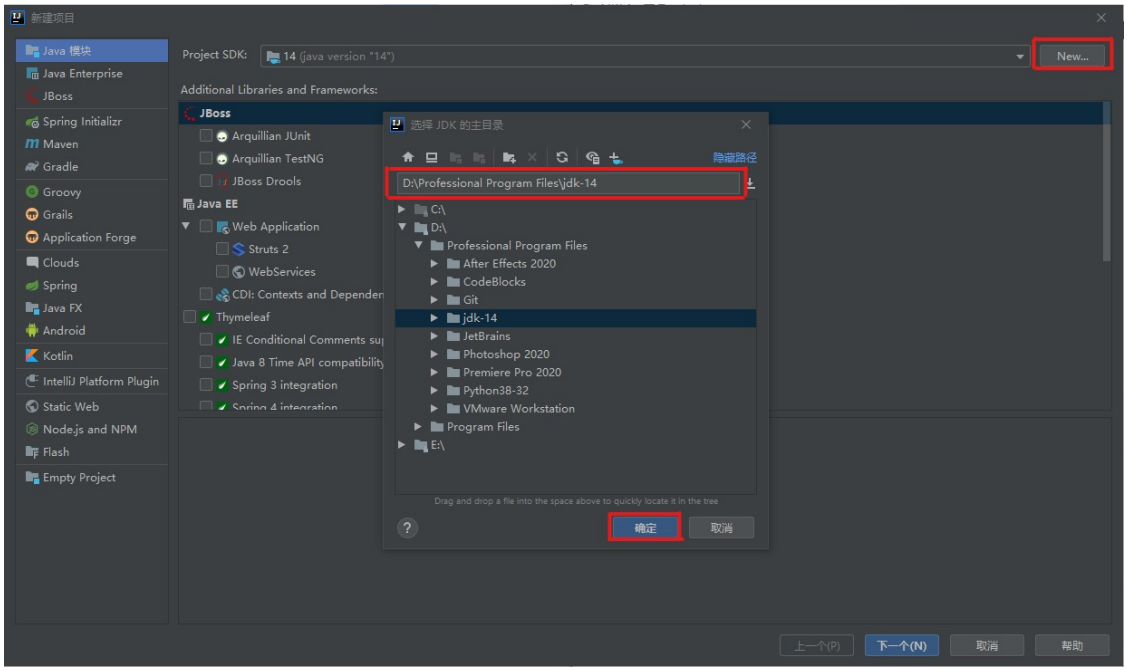
1.1 软件安装

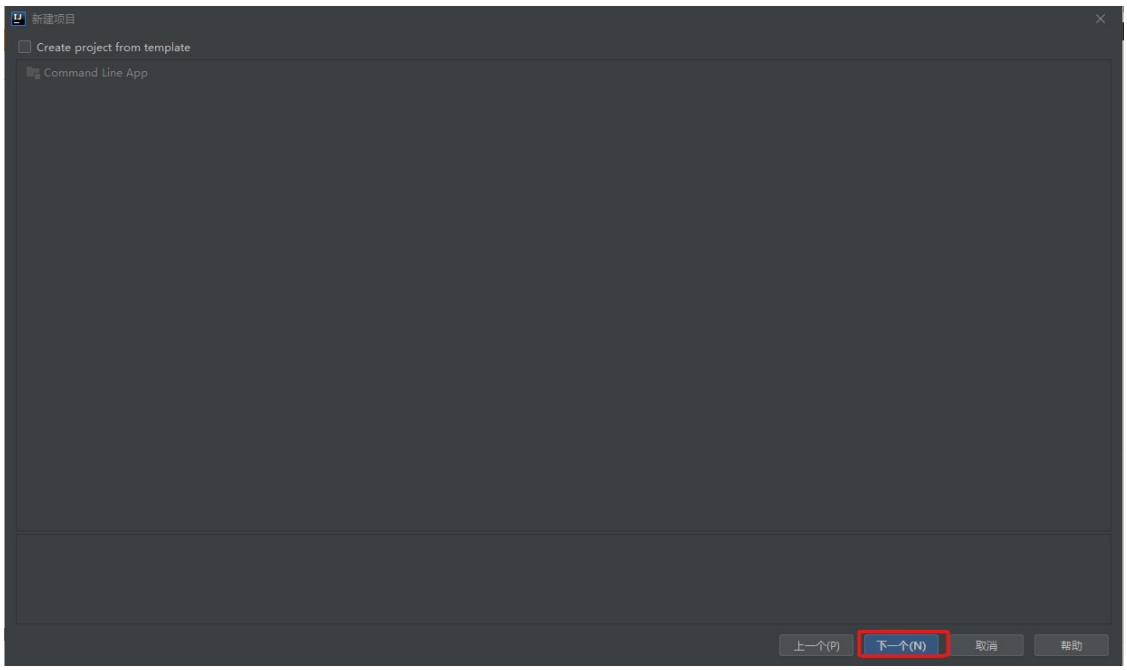
IDEA2019汉化破解版版的详细安装过程：https://mp.weixin.qq.com/s/Gh7oVK2K7X2WY6nKACGXiQ

1.2 创建工程、包和类

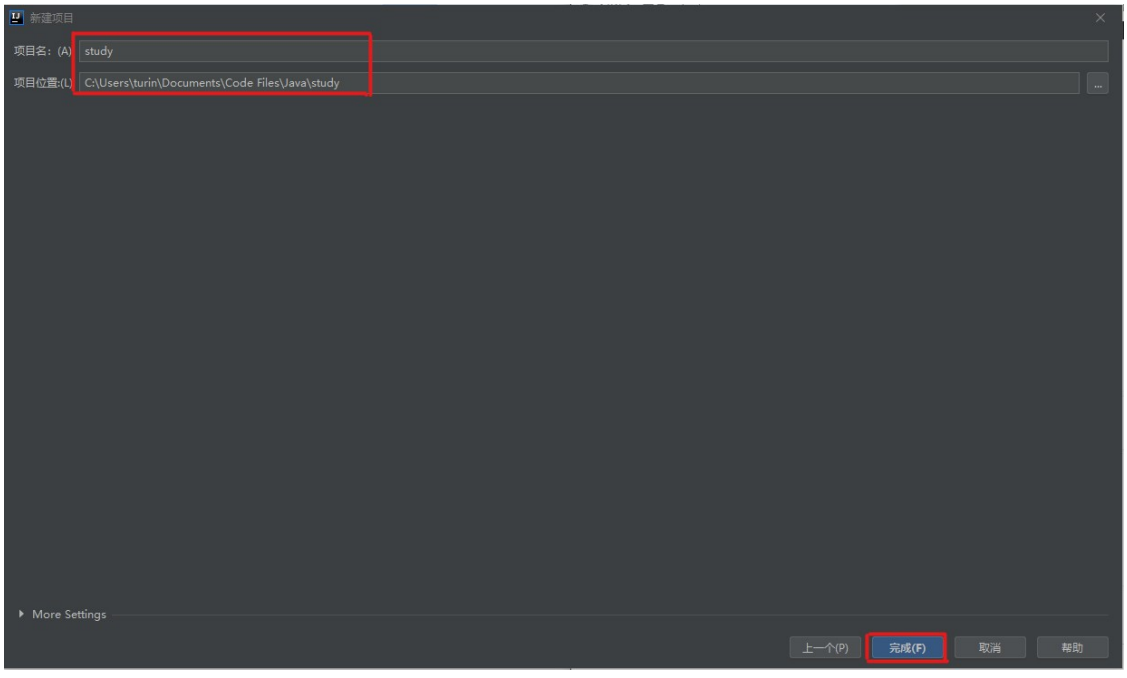
- 软件安装完毕，创建新工程项目

- 添加JDK安装路径

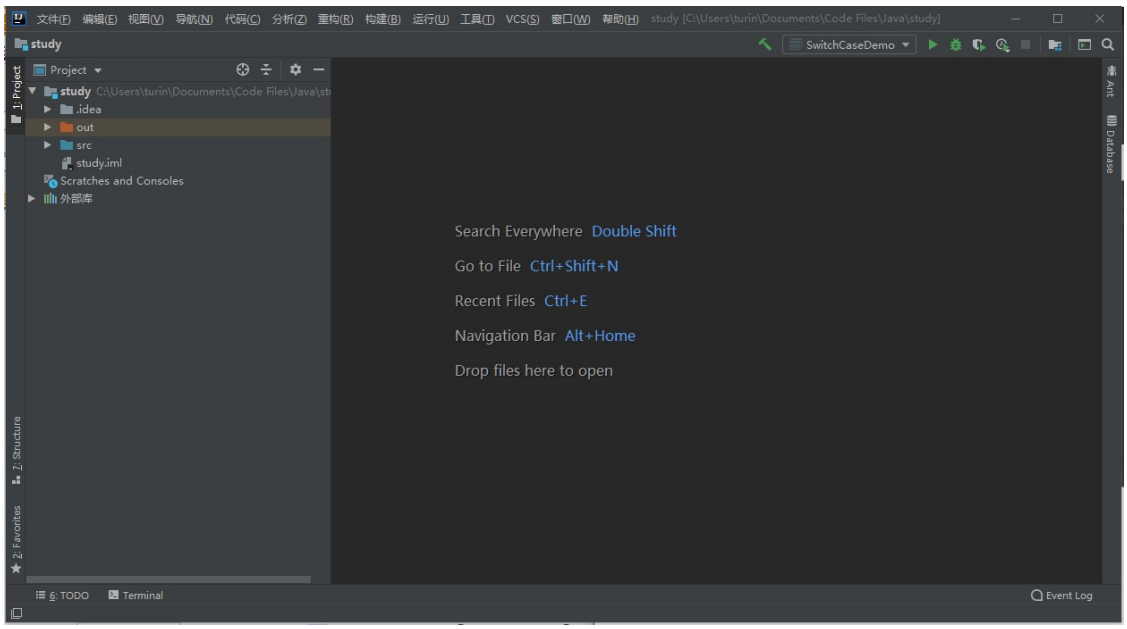




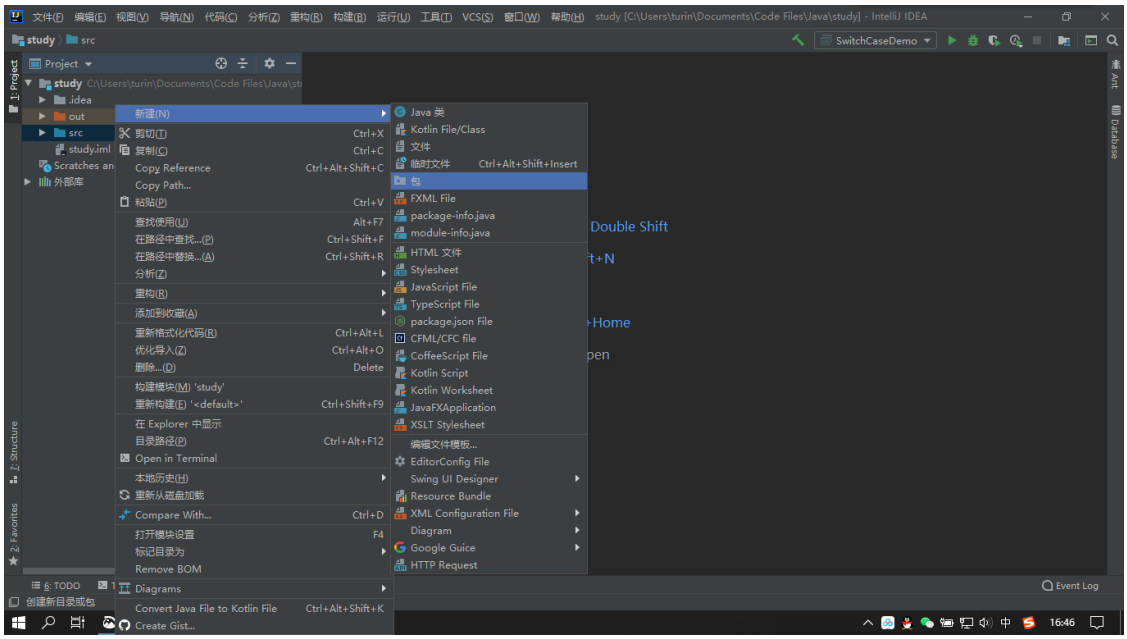
- 修改工程项目名称及文件所在位置

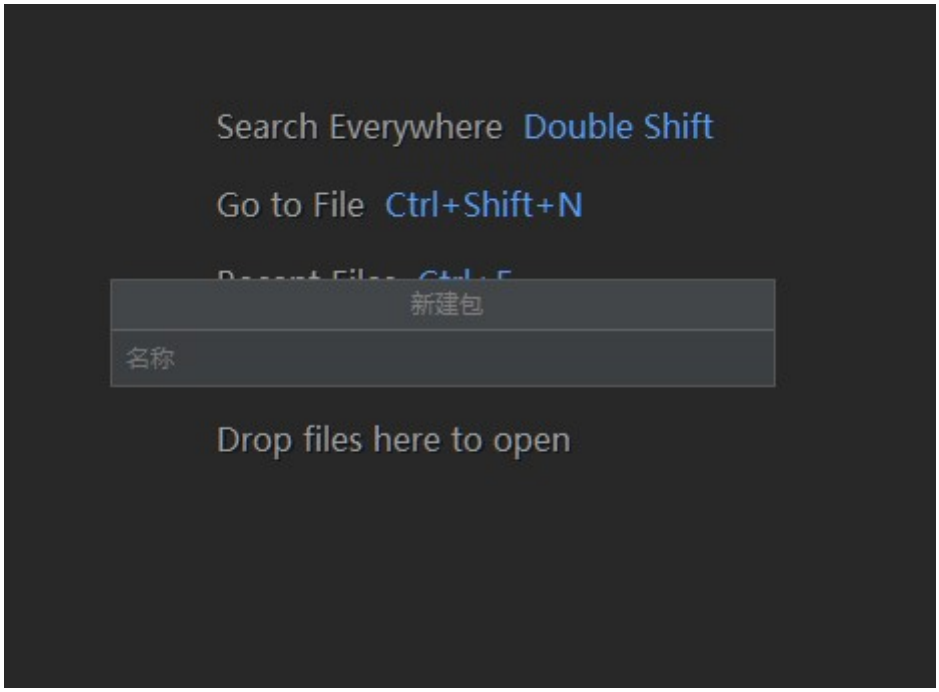


- 创建工程项目完成

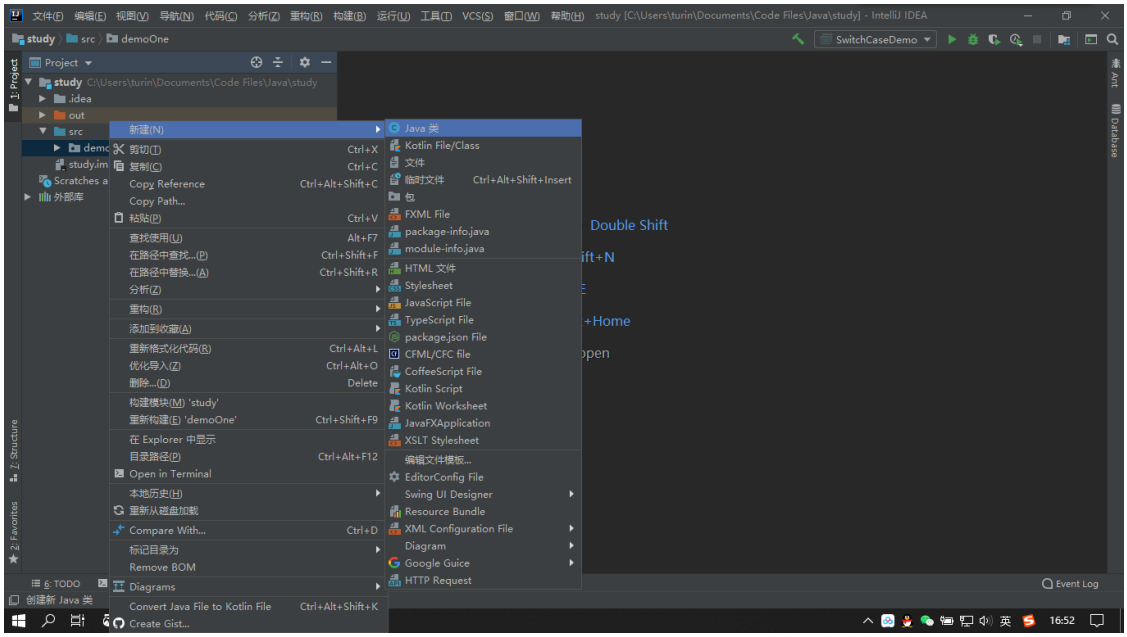


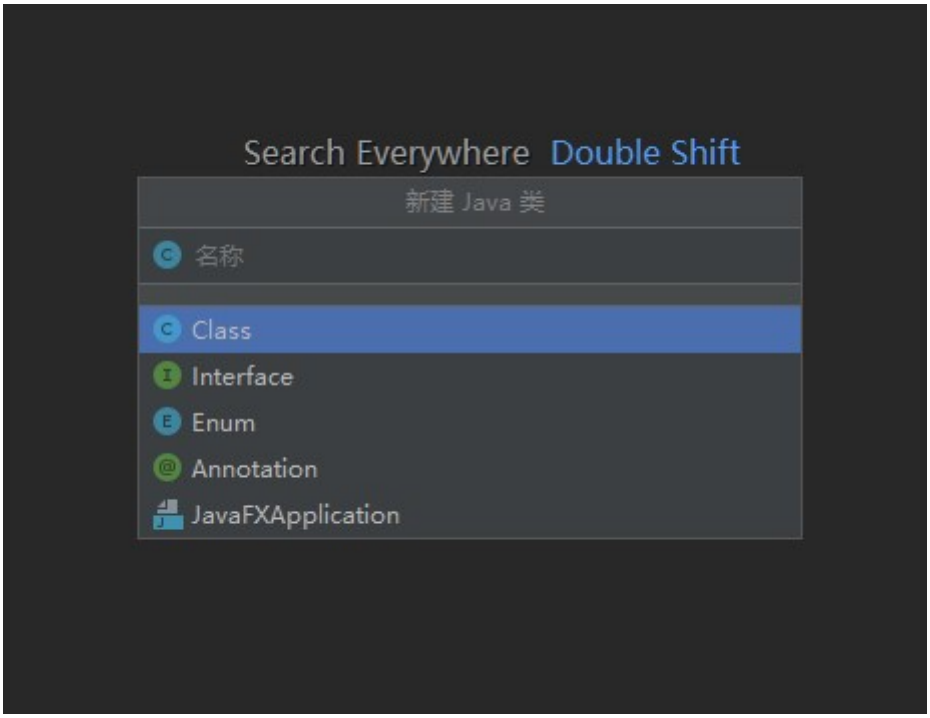
- 创建包



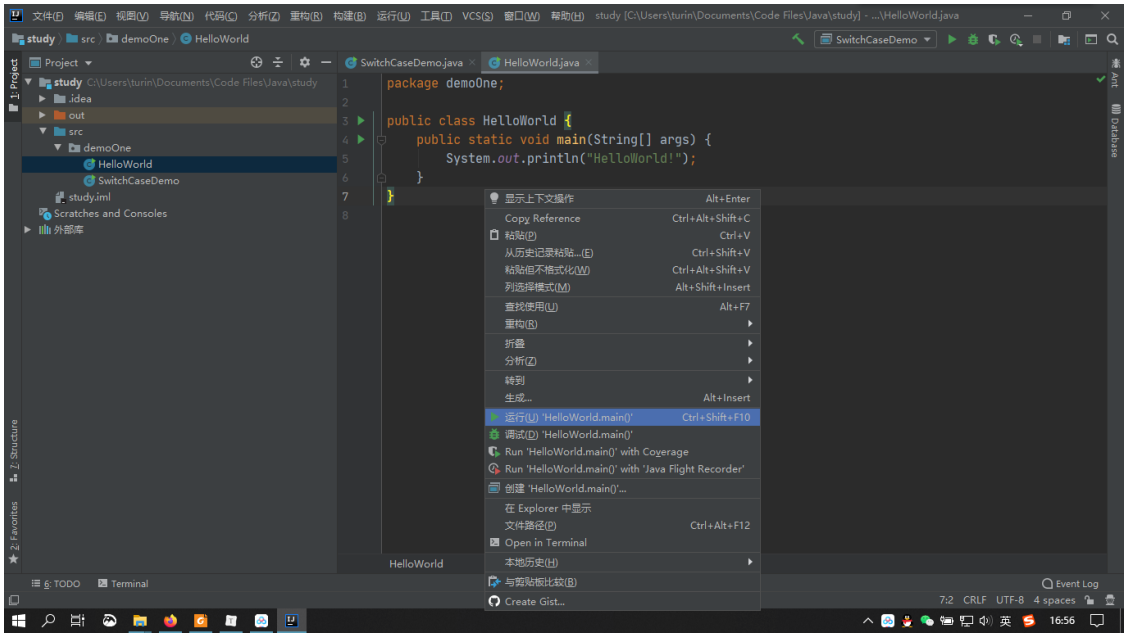


- 创建类





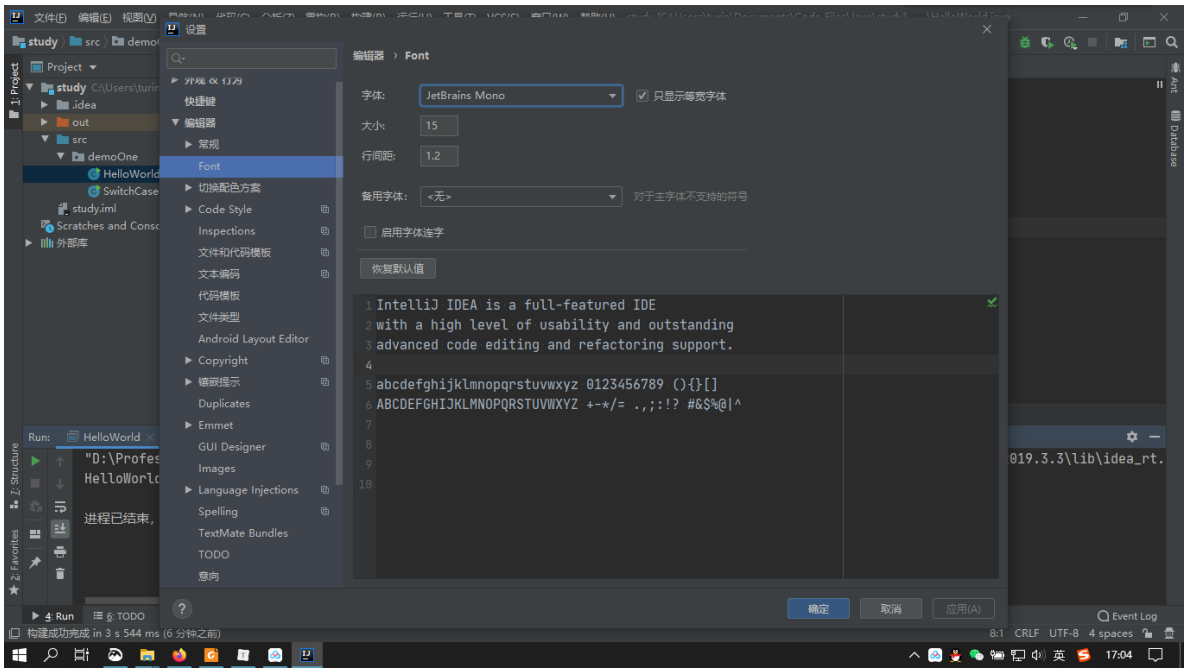
- 在新创建的类中键入代码，右击选择运行，或者通过快捷键`Ctrl`+`Shift`+`F10`运行程序



1.3.1 字体，字号设置

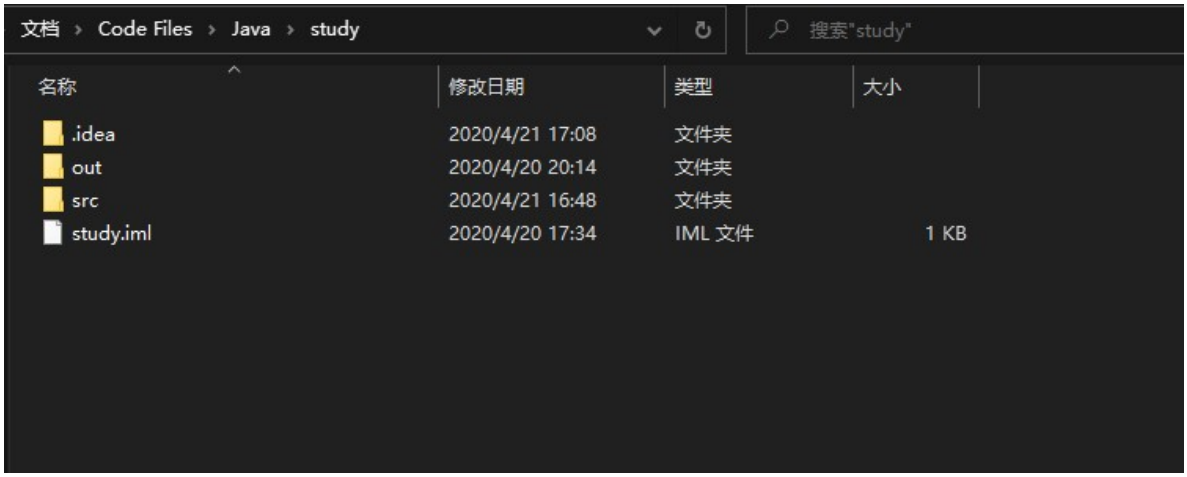
IDEA默认的字体很不友好，字号偏小，建议进行更换

- 点击菜单栏上的`文件 -> 设置 -> 编辑器 -> Font



- 推荐字体使用`JetBrains Mono`，这是`JetBrains`于20年推出的专门为开发人员使用的字体，最新版本的`JetBrains Mono`将从`v2019.3 JetBrains IDE`开始提供。

1.3.2 IDEA的项目目录分析



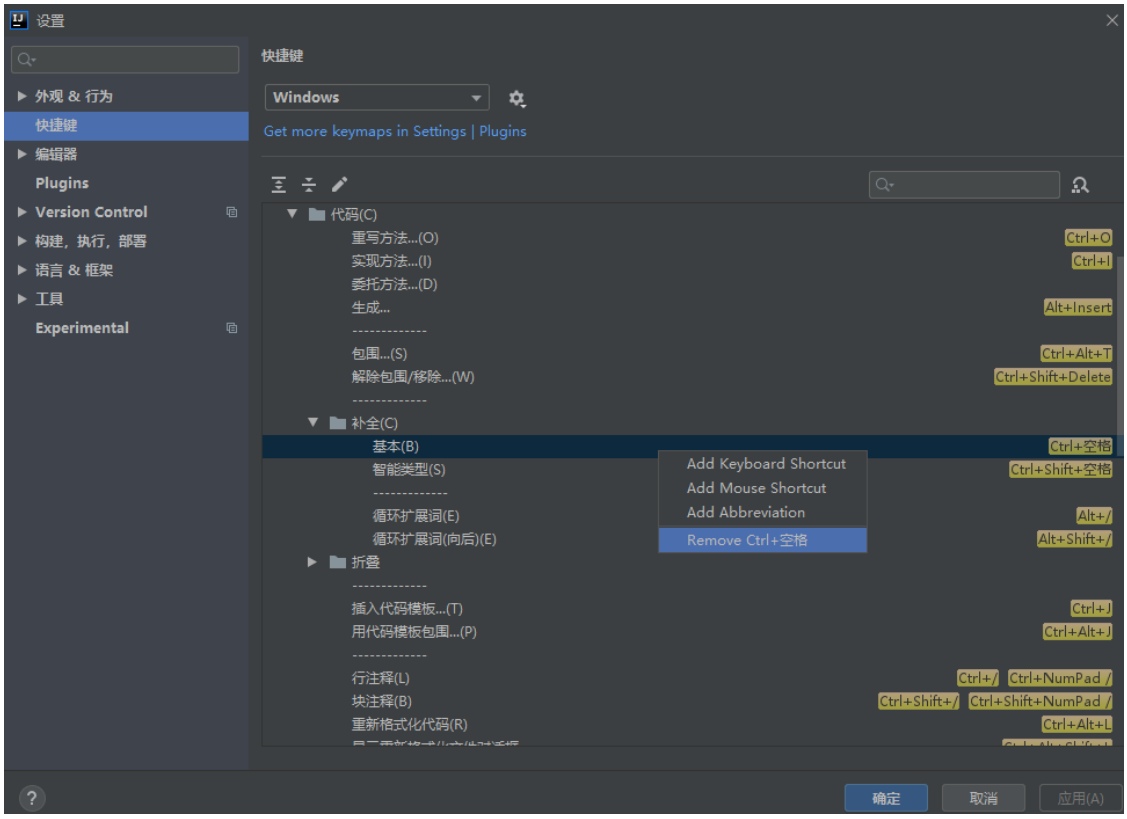
- `.idea`目录与`study.iml`文件和我们开发无关，是IDEA工具自己使用的
- `out`目录是存储编译后的`.class`文件
- `src`目录是存储我们编写的`.java`源文件

1.4 快捷键

快捷键	功能
`Alt+Enter`	导入包，自动修正代码
`Ctrl+Y`	删除光标所在行
`Ctrl+D`	复制光标所在行的内容，插入光标位置下面
`Ctrl+Alt+L`	格式化代码
`Ctrl+/`	单行注释
`Ctrl+Shift+/`	选中代码注释，多行注释，再按取消注释
`Alt+Ins`	自动生成代码，toString，get，set等方法
`Alt+Shift+上下箭头`	移动当前代码行

- 修改快捷键：在IDEA工具中，`Ctrl+Space`的快捷键，可以帮助我们补全代码，但是这个快捷键和Windows中的输入法切换快捷键冲突，需要修改

  `文件 -> 设置 -> 快捷键 -> 主菜单-> 代码 -> 补全 -> 基本`



双击`Remove Ctrl+空格`，再次双击`Add Keyboard Shortcut`，键入`Alt+/`，点击OK

第二章 方法

之前对于方法的使用做了粗略的介绍，这里将对方法进行详解

2.1 方法定义格式的详解

```
修饰符 返回值类型 方法名(参数列表) {
    方法体;
    return 返回值;
}
```

- 修饰符：`public static`固定写法
- 返回值类型：表示方法运行的结果的数据类型，方法执行后将结果返回到调用者
- 参数列表：方法在运算过程中的未知数据，调用者调用方法时传递
- return：将方法执行后的结果带给调用者，方法执行到`return`，整体方法运行结束
- 定义方法需要明确两点
    - **返回值类型**：方法计算的是整数的求和，结果也必然是个整数，返回值类型定义为`int`类型
    - **参数列表**：计算哪两个整数的和，并不清楚，但是可以确定是整数，参数列表可以定义为两个`int`类型的变量，由调用者调用方法时传递
- 注意事项
    - 定义位置：类中方法外
    - 返回值类型，必须和`return`语句返回的类型相同，否则编译失败
    - 不能在`return`后面写代码，`return`意味着方法结束，所有后面的代码永远不会执行，属于无效代码

例：

```
public class Method_Demo {
    public static void main(String[] args) {
        int sum = getSum(5,6);
        System.out.println(sum);
    }
    public static int getSum(int a,int b) {
        return a+b;
    }
}
```

解析：程序执行，主方法`main`调用`getSum`方法，传递了实际数据`5`和`6`，两个变量`a`和`b`接收到的就是实际参数，并将计算后的结果返回，主方法`main`中的变量`sum`接收的就是方法的返回值

2.2 调用方法的三种形式

- 直接调用：直接写方法名调用
- 赋值调用：调用方法，在方法前面定义变量，接收方法返回值
- 输出语句调用：在输出语句中调用方法，`System.out.println(方法名());`
    - 没有返回值的方法不能用输出语句调用

2.3 定义方法的练习

2.3.1 比较两个整数是否相同

```
public class MethodOne {
    public static void main(String[] args) {
        boolean bool=compare(3,4);
        System.out.println(bool);
    }
    /**
     * 定义比较两个整数是否相同的方法
     * 返回值类型：比较结果的布尔类型
     * 参数：两个整数
     * */
    public static boolean compare(int a,int b) {
        if(a==b){
            return true;
        }else{
            return false;
        }
    }
}
```

2.3.2 计算1+2+3+...+100的和

```
public class MethodTwo {
    public static void main(String[] args) {
        int sum = getSum();
        System.out.println(sum);
    }
    /**
```

```
     * 定义计算1~100的求和公式
     * 返回值类型：计算结果整数int
     * 参数：没有参数
     * */
    public static int getSum() {
        int sum = 0;
        for (int i = 1; i <= 100; i++) {
            sum += i;
        }
        return sum;
    }
}
```

2.3.3 实现不定次数打印

```
public class MethodThree {
    public static void main(String[] args) {
        printHelloWorld(6);
    }
    /**
     * 定义打印HelloWorld方法
     * 返回值类型：没有返回值，void
     * 参数类型：整数
     * */
    public static void printHelloWorld(int n) {
        for (int i = 0; i < n; i++) {
            System.out.println(i+1 + "：HelloWorld");
        }
    }
}
```

2.4 方法重载

方法重载：指在同一个类中，允许存在一个以上的同名的方法，只要他们的参数列表不同即可，与修饰符和返回值类型无关

- 参数列表：个数不同，数据类型不同，顺序不同
- 重载方法的调用：JVM通过方法的参数列表，调用不同的方法

2.5 方法重载的练习

2.5.1 比较两个数据是否相等

```
public class MethodFour {
    public static void main(String[] args) {
        //定义不同数据类型的变量
        byte a = 10;
        byte b = 20;
        short c = 10;
        short d = 20;
        int e = 10;
        int f = 10;
        long g = 20;
        long h = 20;
```

```
        //调用
        System.out.println(compare(a,b));
        System.out.println(compare(c,d));
        System.out.println(compare(e,f));
        System.out.println(compare(g,h));
    }
    //两个byte类型
    public static boolean compare(byte a,byte b) {
        System.out.print("byte:");
        return a==b;
    }
    //两个short类型
    public static boolean compare(short a,short b) {
        System.out.print("short:");
        return a==b;
    }
    //两个int类型
    public static boolean compare(int a,int b) {
        System.out.print("int:");
        return a==b;
    }
    //两个long类型
    public static boolean compare(long a,long b) {
        System.out.print("long:");
        return a==b;
    }
}
```

2.5.2 模拟输出语句`println`方法

```
public class MethodFive {
    public static void println(byte a) {
        System.out.println(a);
    }

    public static void println(short a) {
        System.out.println(a);
    }

    public static void println(int a) {
        System.out.println(a);
    }

    public static void println(long a) {
        System.out.println(a);
    }

    public static void println(float a) {
        System.out.println(a);
    }

    public static void println(double a) {
        System.out.println(a);
    }

    public static void println(char a) {
```

```
        System.out.println(a);
    }

    public static void println(String a) {
        System.out.println(a);
    }
    public static void println(boolean a) {
        System.out.println(a);
    }
}
```