АБСТРАКТНЫЕ КЛАССЫ

В TS существует возможность создать абстрактные классы - **концепт, некий шаблон чего-то.** Таким образом в нем могут находится и шаблонные методы-типы, которые должны присутствовать и мы их должны будем еще создать и готовые, которые сразу можно пустить в работу:

```ts
abstract class AbstractVehicle {
    model: string;
    capacity: number;
    abstract startEngine: (time: Date) => string;
    stopEngine(time: Date): string {
        this.startEngine(new Date());
        return 'Engine Stopped'
    }
}

class Vehicle extends AbstractVehicle {
    startEngine = (time: Date) => {
        return 'Started'
    };
}
```

Метод, указанный через abstract должен быть реализован у потомка в соответствии с заданным типом. А метод stopEngine **уже готов к использованию**, как и два свойства. Готовые методы **невозможно** было бы передать при имплементации интерфейса, ведь в нем содержатся только типы:

```ts
interface IEngine {
    model: string;
    capacity: number;
    startEngine: (time: Date) => string;
}

class Vehicle implements IEngine {
    model: string;
    capacity: number;
    startEngine = (time: Date) => {
        return 'Started'
    };
}
```

Так что абстрактные классы могут дать немного больше свободы. У них есть два **ограничения**:

- 👉 У них **нельзя создать экземпляры**, только отнаследоваться от них
- 👉 Нельзя создавать **отдельно абстрактные методы** без объявления всего класса абстрактным. Получится так, что экземпляр класса создать можно, не неизвестно чем будет абстрактный метод. Так что это запрещено

  После компиляции сами классы-абстракции остаются, но без абстрактных методов. Они появляются только у потомков