2022/07/21 追加課題 村上航暉

課題 4-1

> 以下の各関数の引数と戻り値に、型アノテーションを追加してください。

1.

```
function calc(a, b) {
  return a + b;
}

console.log(calc(1));    // 2
console.log(calc(2, 3)); // 5
```

.\src\4-1-1.ts にて実装。

```
function calc(a: number, b = 1): number
```

2.

```
function reply(replyType) {
  switch (replyType) {
    case "Accepted":
      return 0;
      break;
    case "Declined":
      return 1;
      break;
  }
}

console.log("Accepted");  // 正常
console.log("Hoge");      // コンパイルエラー
```

.\src\4-1-2.ts にて実装。

```
type ReplyType = 'Accepted' | 'Declined';

function reply(replyType: ReplyType): number
```

3.

```
function greetings(val) {
  console.log(`${val.message()}, value: ${val.name}`);
}

greetings({
  name: 'aaa',
  message: () => 'hello',
});
```

.\src\4-1-3.ts にて実装。

```
type Message = () => string;
type Greetings = {
  name: string;
  message: Message;
};

function greetings(val: Greetings): void
```

課題 4-2

> 以下は、図形インスタンスを格納する Container クラスの実装（途中まで）です。 Container クラスはジェネリクスを使用して、長方形や円のインスタンスを扱えるようにする予定です。

> 1. Container クラスには型パラメータの指定が不足しています。 Shape とそのサブタイプのみを受け付けるよう、型パラメータ T の宣言を追加してください。

```
/**
 * 図形
 */
interface Shape {
  /**
   * 面積を取得します。
   *
   * @returns {number} 面積
   */
  getArea: () => number;
}

/**
 * 長方形
 */
class Rectangle implements Shape {
  /**
   * @param {number} height 横の辺
   * @param {number} width 縦の辺
   */
  constructor(public readonly height: number, public readonly width: number) {}
```

```
  getArea: () => number = () => this.height * this.width;
}

/**
 * 円
 */
class Circle implements Shape {
  /**
   * @param {number} radius 半径
   */
  constructor(public readonly radius: number) {}
  getArea: () => number = () => Math.pow(this.radius, 2) * Math.PI;
}

/**
 * 図形を格納するコンテナ
 * @template T 図形の型
 */
class Container {
  private shapes: T[] = [];

  /**
   * @param {T} shape 図形
   */
  push(shape: T): void {
    this.shapes.push(shape);
  }

  /**
   * 最後にpushした要素を取得します。
   * @returns {T | undefined} 先頭の要素。空の場合はundefined。
   */
  pop(): T | undefined {
    return this.shapes.pop();
  }
}

const c1 = new Container();
c1.push(new Rectangle(1, 1));  // 正常
c1.push(new Circle(1));        // 正常
c1.push('hoge');               // コンパイルエラー
```

.\src\4-2.ts にて実装。 (次の課題にて変更を行ったため、該当箇所を以下に記載)

```
class Container<T extends Shape>
```

> 2. 型パラメータのデフォルトが Rectangle クラスになるように、Container クラスの型パラメータを宣言してください。

```
const c1 = new Container();
c1.push(new Rectangle(1, 1));
c1.push(new Circle(1));  // コンパイルエラー

const c2 = new Container<Circle>();
c2.push(new Circle(1));
```

`.\src\4-2.ts` にて実装。

```
class Container<T extends Shape = Rectangle>
```

課題 4-3

> 以下は引数に取った型情報のインスタンスを生成して返す Factory クラスの実装です。
> Factory.getInstance の引数 type はどのような型か、回答してください。 参考: https://typescript-jp.gitbook.io/deep-dive/type-system/callable#nyaburunewable

```
/**
 * 単純なファクトリー
 */
class Factory {
  /**
   * @param {string} name 生成時に設定する名称
   */
  constructor(private readonly name: string) {}

  /**
   * 引数に渡された型のインスタンスを生成します。
   *
   * @template 生成する型
   * @param type
   * @returns 型パラメータのインスタンス
   */
  getInstance<T>(type: { new (name: string): T }): T {
    return new type(this.name);
  }
}

const factory = new Factory('test');
console.log(factory.getInstance(TypeA).name);
```

回答: `name`プロパティ(string 型)を持ち、`name`プロパティのみを引数に取るコンストラクタが存在する型

```
class TypeA {
  constructor(public name: string) {
    this.name = name;
```

```
  }
}
```

課題 4-4

> 以下は、連想配列 dictionary の実装（途中まで）です。 dictionary の key は文字列型、value は「number 型の status 属性を持つオブジェクト」を想定しています。
>
> 以下の実装が成り立つように、dictionary の型アノテーションを追加してください。
>
> 参考: https://typescript-jp.gitbook.io/deep-dive/type-system/index-signatures#indekkusushigunechawosuru

```
const dictionary = {};
dictionary['hoge'] = { status: 200 };
```

`.\src\4-4.ts` にて確認コードを実装。

```
const dictionary: { [key: string]: { status: number } } = {};
```