CS214: Advanced Application Development

Lab Session 3: Typescript Basics and OOP review.

```
npm install --global typescript
```

- Open a folder
- Open an editor
- Create main.ts file
- Compile: *tsc main.ts*
- Run: *node main.js*

Interfaces

Sellable

• Represents something that has a name and a price.

• Note: The properties are defined as readonly so that their values cannot be changed once set.

Package<T>

• A generic interface for packaging items, where T is the type of the item to be packaged.

• Methods:

– extract(): T | null

• Removes the contained item (sets it to null) and returns it.

• Returns null if the package is empty.

– pack(item: T): boolean

• Attempts to pack the item into the package if it is empty.

• Returns true if the item was successfully packed; otherwise, returns false.

– isEmpty: boolean

• A property that indicates whether the package is empty (i.e., contains no item).

Wrappable

• An interface that extends Sellable and is used to restrict which items can be put into a Matroschka.

• Since it is an interface, it exists only at compile time and all its members are effectively public.

Classes

Product (Abstract Class)

• An abstract class that implements Sellable and serves as the base for concrete product classes.

CS214: Advanced Application Development

• Attributes:
– name: string
– price: number
• Methods:
– Constructor: Takes a name and a price to initialize the product.
– toString(): string – Returns a string representation in the format:
(name, price)
– abstract describe(): string – An abstract method that forces subclasses to provide their own description.

Mirror (Class)
• A class that extends Product to represent a mirror.
• Attributes:
– width: number (private)
– height: number (private)
• Methods:
– Constructor: Takes width and height; calls the parent constructor with the name "mirror" and a base price of 2.
– get area: number – A property accessor that returns the area (width multiplied by height) in square meters.
– reflect<T>(item: T): T – A generic method that displays the reflected item and returns it.
– describe(): string – Overrides the abstract method to return a description that includes the mirror's area.

Matroschka<T> (Generic Class)
• A generic class that extends Product and implements both Wrappable and Package<T>, where T is constrained to types that extend Wrappable.
• Attributes:
– item: T – The contained item.
• Methods:
– Constructor: Takes an item of type T and calls the parent constructor with the name "Doll" and a price equal to 5 plus the wrapped item's price.
– toString(): string – Returns a string in the format:
(name, price){item}
– extract(): T | null – Removes the contained item (setting it to null) and returns it.
– pack(item: T): boolean – Attempts to pack the item if the package is empty; returns true if successful.
– isEmpty: boolean – A property accessor that returns true if the package is empty (i.e., if the item is null).

– describe(): string – Overrides the abstract method to return a description that includes details about its contained item (or "nothing" if empty).

Box<T> (Generic Class)
• A generic class that implements Package<T> for any T that extends Sellable.
• Attributes:
– item: T – The contained item.
– seal: boolean – A flag indicating whether the box is sealed.
• Methods:
– Constructors:
• A no-argument constructor that sets the item to null and the seal to false.
• An overloaded constructor that takes an item of type T, initializes the box with it, and sets the seal to true.
– extract(): T | null – Removes the item (sets it to null) and sets the seal to false, then returns the removed item.
– pack(item: T): boolean – Attempts to pack the item into the box if it is empty; seals the box and returns true if successful, or returns false if the box already contains an item.
– isEmpty: boolean – A property accessor that returns true if the box is empty (i.e., if the item is null).
– toString(): string – Returns a string in the format:
Box {item} Seal: seal

Testing

```
// ======================================================
// EXAMPLE USAGE (For Lab Practice)
// ======================================================

const mirror = new Mirror(3, 4);
console.log("Mirror area:", mirror.area);
console.log(mirror.describe());
mirror.reflect("sample image");

class WrappableProduct extends Product implements Wrappable {
    constructor(name: string, price: number) {
        super(name, price);
    }
    describe(): string {
        return this.toString();
    }
```

```
    }
    const gift = new WrappableProduct("Gift", 10);
    const matro = new Matroschka(gift);
    console.log(matro.describe());

    const box = new Box<WrappableProduct>(gift);
    console.log(box.toString());
    console.log("Extracted from box:", box.extract());
    console.log("Box after extraction:", box.toString());
```