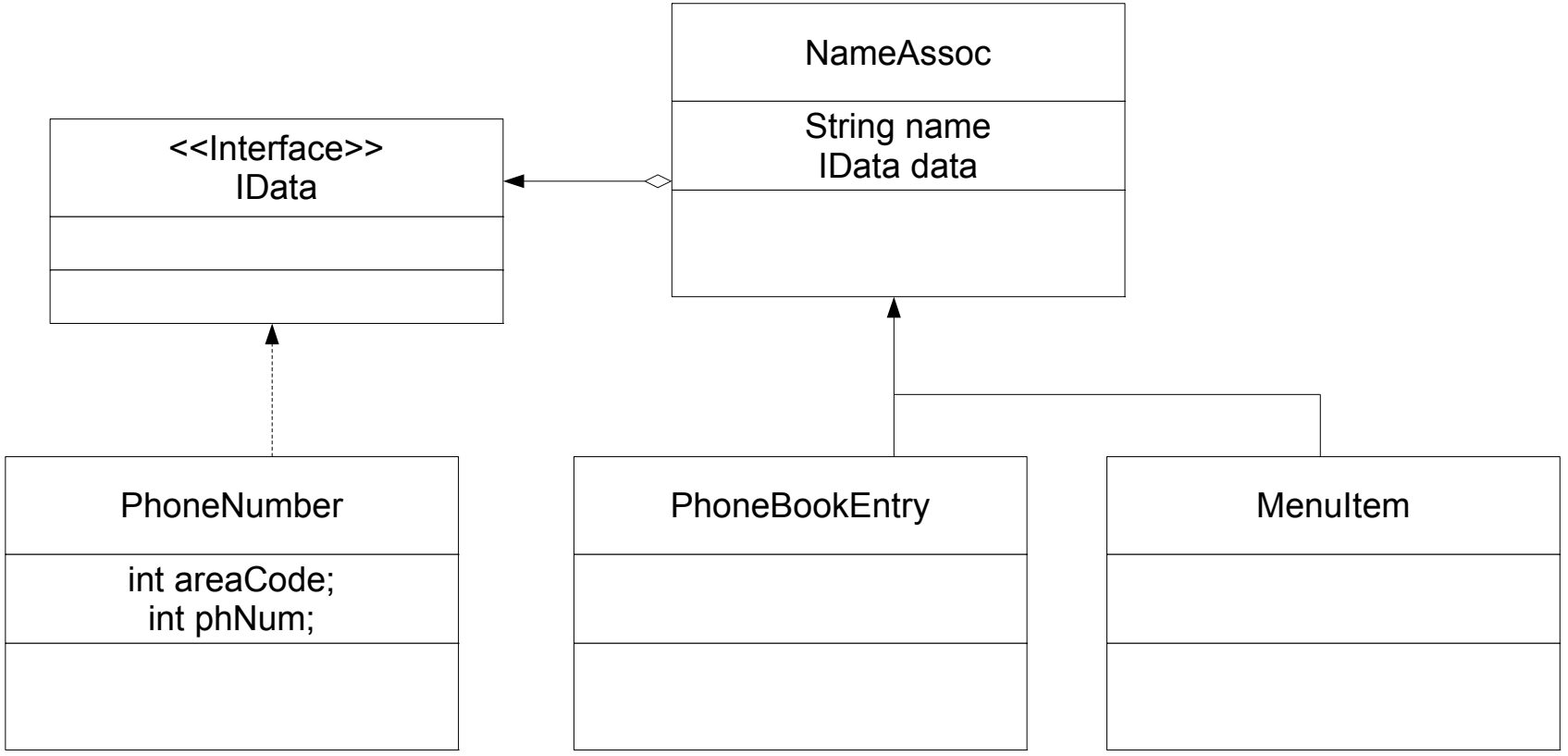
```
class PhoneBookEntry {
  String name;
  PhoneNumber phnum;

  PhoneBookEntry(String name, PhoneNumber phnum) {
    this.name = name;
    this.phnum = phnum;
  }

  void updateNumForName (String forname, PhoneNumber newNum) {
    if (this.name.equals(forname)) {
      this.phnum = newNum;
    }
  }
}

class MenuItem {
  String name;
  int price;

  MenuItem(String name, int price) {
    this.name = name;
    this.price = price;
  }

  void updatePriceForName (String forname, int newPrice) {
    if (this.name.equals(forname)) {
      this.price = newPrice;
    }
  }
}
```

```
class PhoneBookEntry {
  String name;
  PhoneNumber phnum;

  PhoneBookEntry(String name, PhoneNumber phnum) {
    this.name = name;
    this.phnum = phnum;
  }

  void updateNumForName (String forname, PhoneNumber newNum) {
    if (this.name.equals(forname)) {
      this.phnum = newNum;
    }
  }
}

class MenuItem {
  String name;
  int price;

  MenuItem(String name, int price) {
    this.name = name;
    this.price = price;
  }

  void updatePriceForName (String forname, int newPrice) {
    if (this.name.equals(forname)) {
      this.price = newPrice;
    }
  }
}
```

```
class NameAssoc {
    String name;
    ??? data;

    NameAssoc(String name, ??? data) {
      this.name = name;
      this.data = data;
    }

    void updateValForName (String forname, ??? newData) {
      if (this.name.equals(forname)) {
        this.data = newData;
      }
    }
  }
```


NameAssoc
String name
IData data
<<Interface>>
IData
PhoneNumber
int areaCode;
int phNum;
PhoneBookEntry
MenuItem

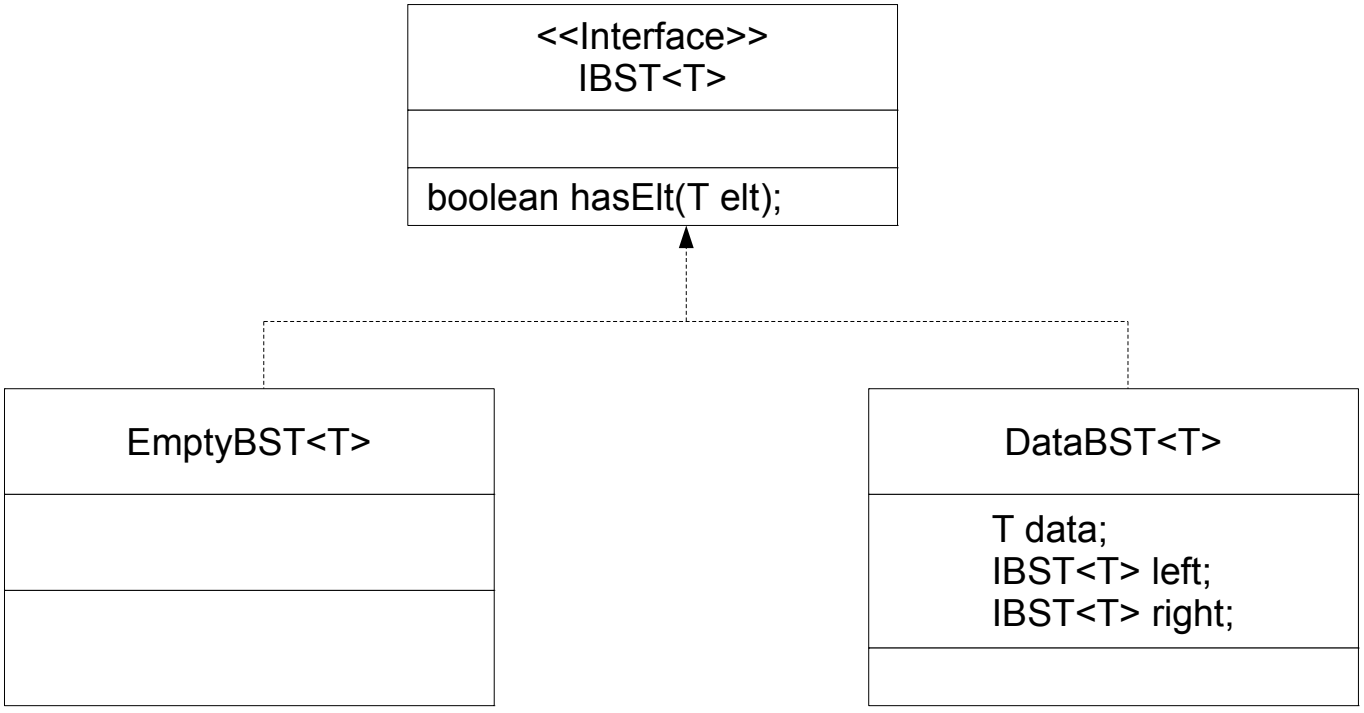
```
class NameAssoc<T> {
        private String name;
        private T data;

        public NameAssoc (String name, T data) {
                …
        }

        public void updateValForName(String forName, T newData) {
                …
        }
}
```

```
class NameAssoc<T> {
        private String name;
        private T data;

        public NameAssoc (String name, T data) {
                …
        }

        public void updateValForName(String forName, T newData) {
                …
        }
}
```

```
class MenuItem extends NameAssoc<Integer> {

        public MenuItem(String name, int price) {
                super(name, price);
        }
}
```

```
Class Pair<S, T> {
        private S data1;
        private T data2;

        public Pair(S d1, T d2) {
                this.data1 = d1;
                this.data2 = d2;
        }
}
```

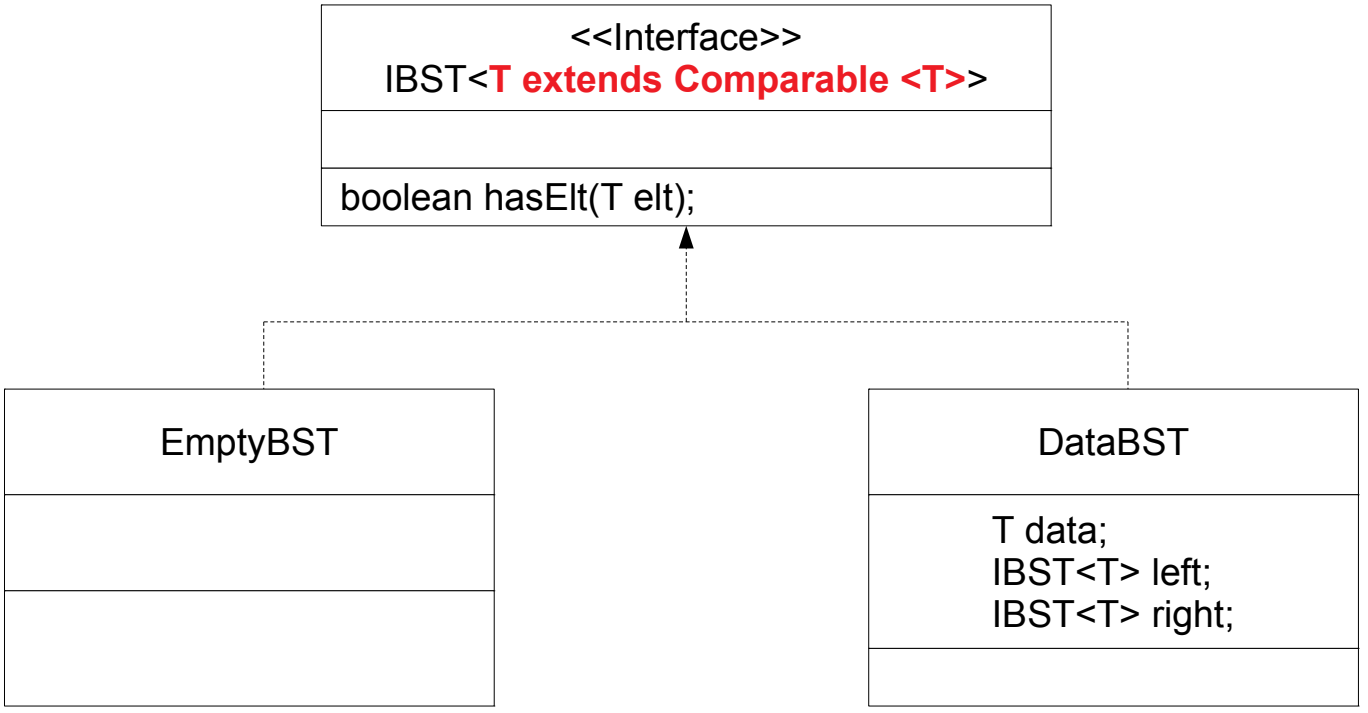
```
Pair <Integer, String> p1 = new Pair<Integer, String>(4, “Hello”);
```


<<Interface>>
IBST<T>
boolean hasElt(T elt);
EmptyBST<T>
DataBST<T>
T data;
IBST<T> left;
IBST<T> right;

```
//write hasElt for a BST that contains integerse
public boolean hasElt(int elt) {
	if(this.data == elt)
		return true;
	else if (elt < this.data)
		return this.left.hasElt(elt);
	else
		return this.right.hasElt(elt);
}
```

```
class Book implements Comparable<Book> {

        ...

        public int compareTo(Book other) {
                if(this.isbn < other.isbn)
                        return -1;
                else if (this.isbn > other.isbn)
                        return 1;
                else
                        return 0;
        }
}
```

```
//write hasElt for a BST that contains integerse
public boolean hasElt(T elt) {
	if(this.data.compareTo(elt) == 0)
		return true;
	else if (this.data.compareTo(elt) < 0)
		return this.left.hasElt(elt);
	else
		return this.right.hasElt(elt);
}
```




```
interface IBST<T extends Comparable<T>> {
        boolean hasElt(T elt);
}
```