

# Exercise Set: Iteration Abstraction

In this exercise set, we have marked questions we think are harder than others with a [‡]. We have also marked questions for which solutions are provided at the end of the set ([SP]). To check solutions for other questions than those marked with [SP], ask one of the instructors or TAs or post a question to Piazza!

1. What are the three different kinds of abstraction used in the Java programming language that we have seen so far this term? Explain each of them.

2. Assume you have a field `fPeople` that is of type `List<Person>` and the type `Person` has a method `String getName()`. The type `List<E>` has a method `iterator()` that returns an object of type `Iterator` as described in lecture. How would you write code that generates a `String` that consists of the names of all people in `fPeople` each separated by a comma? (If possible, think of one solution with, and one without using the method `iterator()` explicitly.) [‡],[SP]

3. Given the following code:

- a) Do you recognize the Iterator design pattern? Briefly discuss the main types and methods of the Iterator design pattern in the following code.
- b) There are several places where a loop would be applied in this code to iterate over a collection of elements. Which methods do you think use a loop in their implementation? [‡],[SP]
- c) How would you write the code for the iteration in the methods you identified in b)?

```
public class AddressBook {
    private Set<Contact> fContacts;

    // Creates an address book with an empty set of contacts.
    AddressBook() {...}

    // Adds a contact to the address book.
    void addContact(Contact c) {...}

    // Removes a contact from the address book if existent.
    void removeContact(String name) {...}

    // Returns the contact with the specified name if existent
    // in the address book.
    Contact getContact(String name) {...}

    // Resets the phone number of all contacts to 0
    void resetAllPhoneNumbers() {...}
}
```

```

public class Contact {
    ...
    // Creates a contact with the specified name
    Contact(String name) {...}

    // Sets the telephone number of this contact
    void setTelephoneNumber(int number) {...}

    // Returns the name of this contact
    String getName() {...}
}

public class Set {
    ...
    // Adds the specified element to this set if it is not already present.
    boolean add(Object o) {...}

    // Removes the specified element from this set if it is present.
    boolean remove(Object o) {...}

    // Returns an iterator over the elements in this set.
    Iterator iterator() {...}

    // Returns true if this set contains no elements.
    boolean isEmpty() {...}
}

public class SetIterator implements Iterator {
    // Returns true if the iteration has more elements.
    boolean hasNext() {...}

    // Returns the next element in the iteration.
    Object next() {...}

    // Removes from the underlying collection the last element
    // returned by the iterator.
    void remove() {...}
}

```

## SOLUTIONS:

2. Below is one solution that uses the method `iterator()` explicitly and two that don't. The last one is only different in that it doesn't create a new `String` for every iteration but only appends it to a buffer.

```
// First solution
String allNames = null;
Iterator<Person> anIterator = fPeople.iterator();
while (anIterator.hasNext()) {
    Person aPerson = anIterator.next();
    if (allNames == null) {
        allNames = aPerson.getName();
    } else {
        allNames = allNames + "," + aPerson.getName();
    }
}
```

```
// Second solution
String allNames = null;
for (Person person : fPeople) {
    if (allNames == null)
        allNames = person.getName();
    else
        allNames = allNames + "," + person.getName();
}
```

```
// Third solution
StringBuffer buffer = new StringBuffer();
for (Person person : fPeople) {
    if (buffer.length() == 0) {
        buffer.append(person.getName());
    } else {
        buffer.append(",");
        buffer.append(person.getName());
    }
}

String allNames = buffer.toString();
```

3. b) The implementation of each of the three methods in type `AddressBook` listed below would use a loop to iterate over the contacts:

```
void removeContact(String name) {...},
Contact getContact(String name) {...},
void resetAllPhoneNumbers() {...}
```