Exercise 7 – Collections and Generics

Objective

This exercise looks at how to use collections and generics as well as write some more of our own simple objects. By the end of the exercise you should be more comfortable with creating and using collections.

Overview

Project Setup

- Create a new java project and the package “com.qa”
- Inside the com.qa package create the file main.java with a main method. This method will be where your code is later
- Create another package, “com.qa.model”
- Inside this package create four java files with the following code:

Animal.java

```java
package com.qa.model;

public abstract class Animal {
	//private fields
	protected String name;
	private int age;
	//constructor
	public Animal(String name, int age){
		this.name = name;
		this.age = age;
	}
	//setters and getters for name and age
	public void setName(String name) { this.name = name;	}
	public String getName() { return name; }
	public void setAge(int age) { this.age = age;}
	public int getAge() { return age;}
	//abstract methods, the animal class doesn't know
	//how to implement these. We need some concrete
	//instantiations
	public abstract String sayHello();

	public abstract String move();

	public String toString(){
		return "Name: " + name + " Age: " + age;
	}
}
```

Cat.java

```
package com.qa.model;

public class Cat extends Animal{
     public Cat(String name, int age) {
          super(name, age);
     }
     @Override
     public String sayHello() {
          return "Oh. It's you. Hello." + name;
     }
     @Override
     public String move() {
          return "If you insist...";
     }
     public String toString(){
          return "Cat: " + super.toString();
     }
}
```

Dog.java

```
package com.qa.model;


public class Dog extends Animal {
	public Dog(String name, int age) {
		super(name, age);
	}
	@Override
	public String sayHello() {
		return "Oh hello! You're back! Hello! Hello! I missed 
you! Hello!";
	}
	@Override
	public String move() {
		return "Ruuuuuuuuuuun!";
	}
	public String toString(){
		return "Dog: " + super.toString();
	}
}
```

Rabbit.java

```java
package com.qa.model;

public class Rabbit extends Animal {
	public Rabbit(String name, int age) {
		super(name, age);
	}
	@Override
	public String sayHello() {
		return "Snuffle snuffle";
	}
	@Override
	public String move() {
		return "Hop hop hop";
	}
	public String toString(){
		return "Rabbit: " + super.toString();
	}
}
```

Creating collections

1. Create five objects from the classes you have. These should be different animal types.
2. Create an ArrayList containing all the objects
    a. Remember to give it the type parameter using <>
    b. Look at the difference when you give it the sub-type (cat, dog, rabbit) rather than the supertype (animal).
    c. Use the debugger to inspect the object as the animals are being added to it.

3. Create a HashMap of the objects
    a. First, create a HashMap that uses the name of the animal as a key and the object as the value.
    b. We can also use the object itself as the key in a Hashmap. Create a second HashMap that uses the object as the key and a description for value.
4. Create a set of the objects, use whichever set you like - have a look at the api documentation for this
    a. Try adding an object to the set more than once. Will it let you? Look at what is happening to the object using the debugger.
5. Print out the values in your collections. You can do this using a for loop, a for-each loop or the iterator. Try all three methods.
6. Find a specific object in your collections. Chose one of your objects (for example, a cat called “Bob”). Look through the API to see if there are any methods that can help you.

    Hint: For some collections you need to iterate through each items. For others you can just go directly to the object in question.
7. Now we want to sort our List. As this is the only collection that uses an ordered list of elements we can use the sort method to do this.
    a. Sort the contents of your ArrayList by using the Collections.sort() method. This will need the animal class to extend Comparable and implement the compareTo method in either the parent class or each of the children. Sort the animals by their age.
    b. We can’t use the sort() method on a Set or a HashMap. The order of elements in a HashMap is based on the key, rather than the order they were inserted. Look at the Java API and see if there are any other types of maps or sets you can use that will sort the collections for us. Implement these collections and have a look at the output.

Glossary of key terms

Collection

It is a group of java objects

List

It is an interface which concentrates on index and does allow duplication

Set

It is an interface which organises the collection objects based on hashcode. It does not allow duplication

Map

It represents the objects based on the key-value pairs

ArrayList

It is one of the implementation classes of **List** interface and it is not synchronized

HashMap

It is an implementation class of **Map** interface which maintain the objects based on key-value pair. HashMap uses hashcode.

Generic class

This type of class provides type-safety and avoids explicit type casting