同樣是收集物件，在收集過程中若有相同物件，則不再重複收集，若你有這類需求，可以使用Set介面的實作物件。例如，若有一個字串，當中有許多的英文單字，你希望知道不重複的單字有幾個，那麼你可以如下撰寫程式：

```
package cc.openhome;

import java.util.*;

public class WordCount {
    public static void main(String[] args) {
        String line = input("請輸入英文：");
        Set<String> words = tokens(line);
        System.out.printf("不重複單字有 %d 個：%s%n",
                words.size(), words);
    }

    static String input(String prompt) {
        System.out.print(prompt);
        Scanner scanner = new Scanner(System.in);
        String line = scanner.nextLine();
        return line;
    }

    static Set<String> tokens(String line) {
        String[] tokens = line.split(" ");
        Set<String> words = new HashSet<>();
        for(String token : tokens) {
            words.add(token);
        }
        return words;
    }
}
```

String的split()方法，可以指定切割字串的方式，在這邊指定以空白切割，split()會傳回String[]，包括切割的每個字串，接著將String[]中的每個字串加入Set的實作HashSet中，由於Set的特性是不重複，因此若有相同單字，則不會再重複加入，最後只要呼叫Set的size()方法，就可以知道收集的字串個數，HashSet的toString()實作，則會包括收集的字串。一個執行的範例如下：

```
請輸入英文：This is a dog that is a cat where is the student
不重複單字有 9 個：[that, cat, is, student, a, the, where, dog, This]
```

再來看以下的範例：

```
package cc.openhome;

import java.util.*;

class Student {
    private String name;
    private String number;
    Student(String name, String number) {
        this.name = name;
        this.number = number;
    }

    @Override
    public String toString()  {
        return String.format("(%s, %s)", name, number);
    }
```

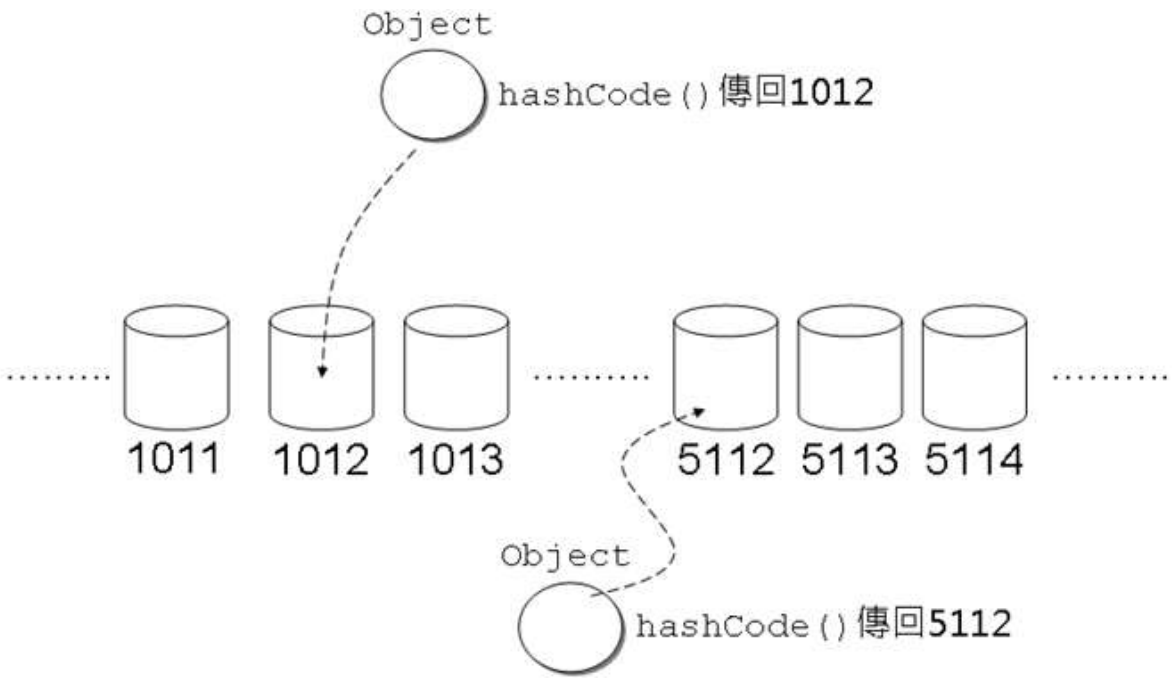
```
}

public class Students {
    public static void main(String[] args) {
        Set<Student> set = new HashSet<>();
        set.add(new Student("Justin", "B835031"));
        set.add(new Student("Monica", "B835032"));
        set.add(new Student("Justin", "B835031"));
        System.out.println(set);
    }
}
```

程式中使用Set收集了Student物件，其中故意重複加入了相同的學生資料，然而在執行結果中看到，Set並沒有將重複的學生資料排除：

```
[(Monica, B835032), (Justin, B835031), (Justin, B835031)]
```

這是理所當然的結果，因為你並沒有告訴Set，什麼樣的Student實例才算是重複，以HashSet為例，會使用物件的hashCode()與equals()來判斷物件是否相同，HashSet的實作概念是，在記憶體中開設空間，每個空間會有個雜湊編碼（Hash code）：



這些空間稱為雜湊桶（Hash bucket），如果物件要加入HashSet，則會呼叫物件的hashCode()取得雜湊碼，並嘗試放入對應號碼的雜湊桶中，如果雜湊桶中沒物件，則直接放入，如上圖所示；如果雜湊桶中有物件呢？會再呼叫物件的equals()進行比較：

如果同一個雜湊桶中已有物件，呼叫該物件equals()與要加入的物件比較結果為false，則表示兩個物件非重複物件，可以收集，如果是true，表示兩個物件是重複物件，則不予收集。

事實上不只有HashSet，Java中許多要判斷物件是否重複時，都會呼叫hashCode()與equals()方法，因此規格書中建議，兩個方法必須同時實作。以先前範例而言，若實作了hashCode()與equals()方法，則重複的Students將不會被收集：

```
package cc.openhome;

import java.util.*;

class Student {
    private String name;
    private String number;
    Student(String name, String number) {
        this.name = name;
        this.number = number;
    }

    // NetBeans自動產生的equals()與hashCode()
    // 就示範而言已經足夠了

    @Override
    public int hashCode() {
        // Objects 有 hash() 方法可以使用
        // 以下可以簡化為 return Objects.hash(name, number);
        int hash = 7;
        hash = 47 * hash + Objects.hashCode(this.name);
        hash = 47 * hash + Objects.hashCode(this.number);
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
```

```
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final Student other = (Student) obj;
        if (!Objects.equals(this.name, other.name)) {
            return false;
        }
        if (!Objects.equals(this.number, other.number)) {
            return false;
        }
        return true;
    }

    @Override
    public String toString()  {
        return String.format("(%s, %s)", name, number);
    }
}

public class Students {
    public static void main(String[] args) {
        Set<Student> set = new HashSet<>();
        set.add(new Student("Justin", "B835031"));
        set.add(new Student("Monica", "B835032"));
        set.add(new Student("Justin", "B835031"));
        System.out.println(set);
    }
}
```

在這邊定義學生的姓名與學號相同，表示為相同Student物件，hashCode()則直接利用Objects的hashCode()再作運算（為NetBeans自動程式碼產生的結果）。執行結果如下，可看出不再收集重複的Students物件：

```
[(Justin, B835031), (Monica, B835032)]
```