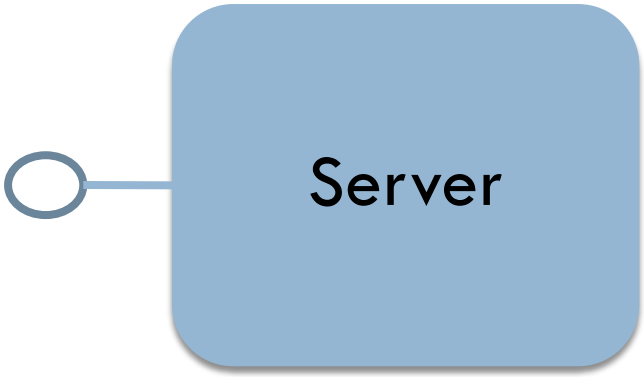
JAVA SERVERSOCKET

030523313 - Network programming

Asst. Prof. Dr. Choopan Rattanapoka

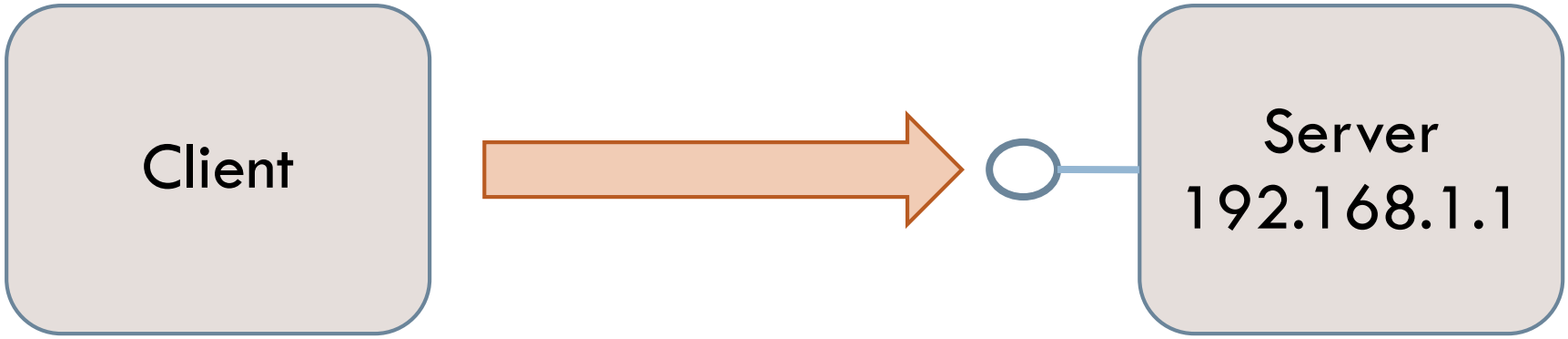
Java ServerSocket



- Server มีหน้าที่รอการติดต่อจาก Client ใน port ที่ตัวเองจะให้บริการ
- ใน Java จะสามารถทำได้โดยการสร้าง Object ของ ServerSocket
- **ตัวอย่าง :**

ServerSocket server = new ServerSocket(หมายเลข port);

Client and Server




```
ServerSocket  server = new ServerSocket (2000);
```

```
Socket  s = new Socket ("192.168.1.1", 2000);
```

ฝ่าย **Server** จำเป็นจะต้องมี **Socket** เพื่อติดต่อกับ **Client** สามารถทำได้โดย

```
Socket  client  =  server.accept();
```

โปรแกรมจะค้างที่บรรทัดนี้จนกว่าจะมี Client มาร้องขอการเชื่อมต่อ

ตัวอย่างโปรแกรมของฝั่ง server

```
1  import java.io.*;
2  import java.net.*;
3
4  public class Server {
5      public static void main(String[] args) {
6          ServerSocket server = null;
7          try {
8              server = new ServerSocket(2000);
9          } catch (Exception e) {
10             e.printStackTrace();
11         }
12
13         while(true) {
14             try {
15                 Socket client = server.accept();
16
17                 //CODE HERE
18
19                 client.close();
20
21             } catch(Exception e) {
22                 e.printStackTrace();
23             }
24         }
25     }
26 }
```

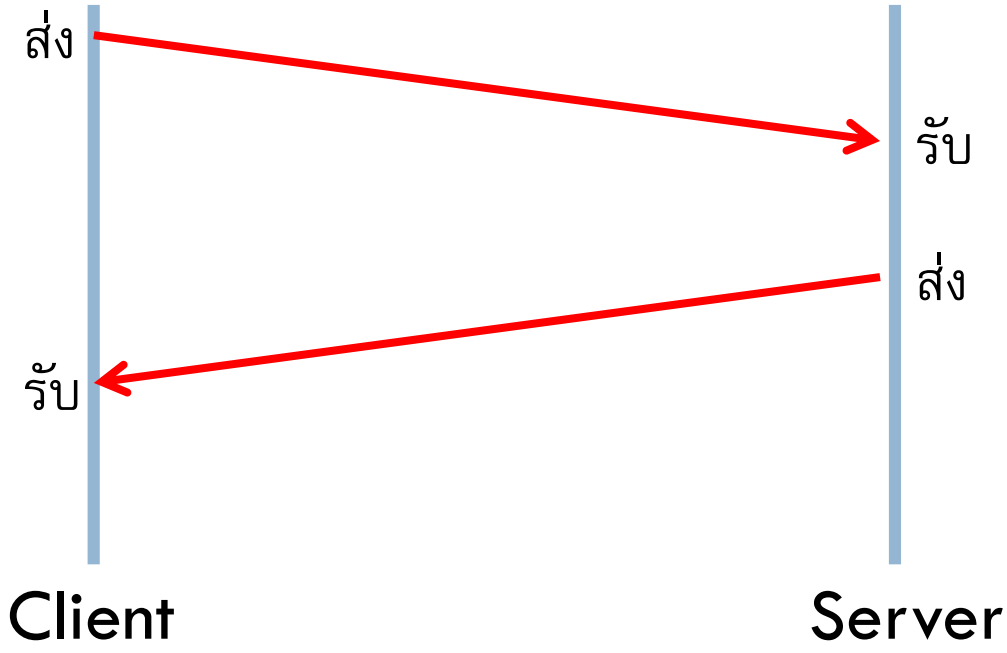
เอา **socket** ออกมาใช้งาน

วน **loop** รอรับการติดต่อจาก **client**

ปิดเมื่อใช้งานเรียบร้อย

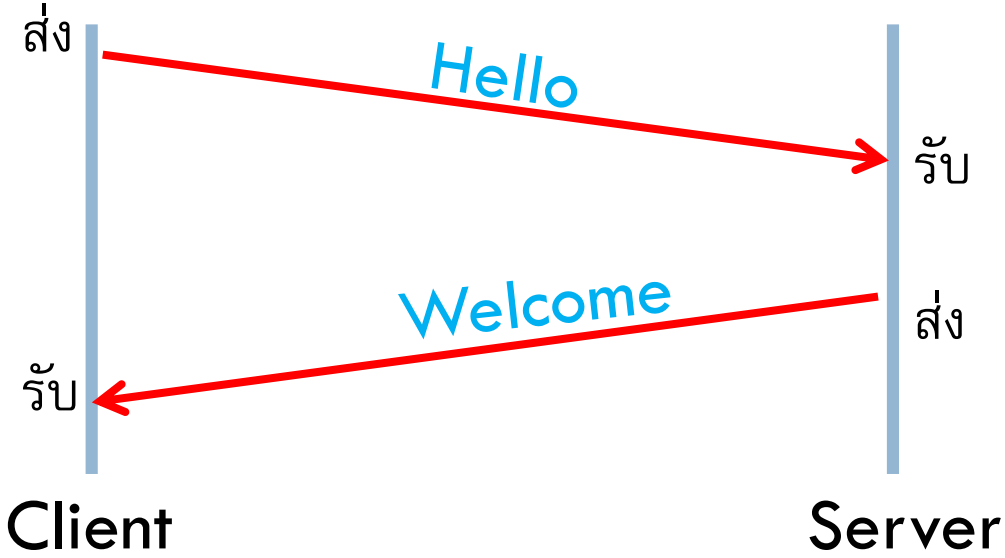
การเขียน server/client

- การเขียน **server** และ **client** ด้วยตัวเองนั้นจะต้องออกแบบ **protocol**
- คือการออกแบบช่วงไหนจะรับข้อมูล ช่วงไหนจะส่งข้อมูล โดย **server** และ **client** จะต้องทำหน้าที่สลับกัน



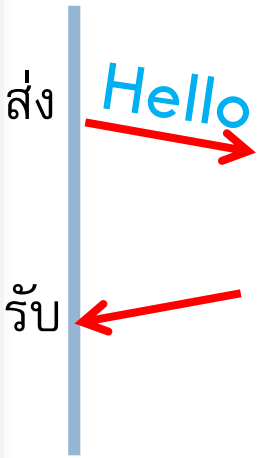
ตัวอย่าง 1: โปรแกรม server/client

- ถ้าต้องการเขียนโปรแกรมให้
  - Server ให้บริการที่ port หมายเลข 10000
  - Client ติดต่อกับ Server พร้อมทั้งส่งคำว่า Hello
  - เมื่อ Server ได้รับคำว่า Hello, server จะตอบคำว่า Welcome
  - จากนั้นปิดการเชื่อมต่อ



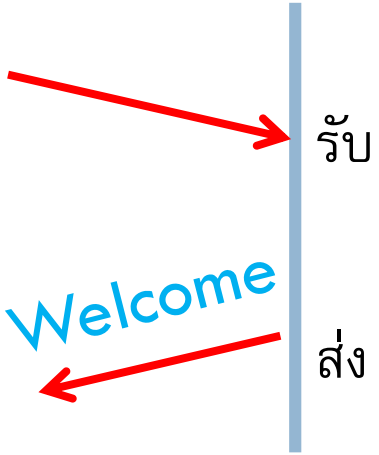
โปรแกรม Client

```
import java.net.*;
import java.io.*;
public class Client {
        public static void main(String[] args) {
                try {
                        Socket s = new Socket("127.0.0.1", 10000);
                        BufferedReader br = new BufferedReader(
                                                new InputStreamReader(
                                                s.getInputStream()));
                        PrintWriter pw = new PrintWriter(s.getOutputStream());

                        pw.println("Hello");
                        pw.flush();

                        String msg = br.readLine();

                        System.out.println(msg);

                        br.close();
                        pw.close();
                        s.close();
                } catch (Exception e) { e.printStackTrace(); }

        }
}
```



โปรแกรม Server

```
import java.io.*;
import java.net.*;

public class Server {
        public static void main(String[] args) {
                ServerSocket servSocket = null;
                try {
                        servSocket = new ServerSocket(10000);
                } catch(Exception e) {e.printStackTrace();}

                while(true) {
                        try {
                                Socket s = servSocket.accept();

                                BufferedReader br = new BufferedReader(
                                                        new InputStreamReader(
                                                        s.getInputStream()));
                                PrintWriter pw = new PrintWriter(
                                                        s.getOutputStream());
                                String msg = br.readLine();

                                pw.println("Welcome");
                                pw.flush();
                                pw.close();
                                br.close();
                                s.close();
                        } catch(Exception e) {}
                }
        }
}
```



ตัวอย่าง 2 : โปรแกรม login, password

- เขียนโปรแกรมให้ server เป็นตัวตรวจสอบ login และ password อย่างง่าย โดยมีการทำงานดังต่อไปนี้
  - Server ให้บริการที่ port 20000
  - Client จะเชื่อมต่อไปยัง Server จากนั้นจะส่ง
    - Login (มาจาก parameter ตัวที่ 1)
    - Password (มาจาก parameter ตัวที่ 2)
  - Server ตรวจสอบว่า login = admin และ password = admin หรือไม่
    - ถ้าใช่ให้ส่งข้อความ OK กลับไปยัง client
    - ถ้าไม่ใช่ให้ส่งข้อความ NO กลับไปยัง client

โปรแกรม Client

```
import java.net.*;
import java.io.*;
public class Client2 {
	public static void main(String[] args) {
		try {
			Socket s = new Socket("127.0.0.1", 20000);
			BufferedReader br = new BufferedReader(
					new InputStreamReader(
					s.getInputStream()));
			PrintWriter pw = new PrintWriter(s.getOutputStream());

			pw.println(args[0]);
			pw.println(args[1]);
			pw.flush();

			String msg = br.readLine();

			System.out.println(msg);

			br.close();
			pw.close();
			s.close();
		} catch (Exception e) { e.printStackTrace(); }

	}
}
```

โปรแกรม Server

```
import java.io.*;
import java.net.*;

public class Server2 {
	public static void main(String[] args) {
		ServerSocket servSocket = null;
		try {
			servSocket = new ServerSocket(20000);
		} catch(Exception e) {e.printStackTrace();}

		while(true) {
			try {
				Socket s = servSocket.accept();

				BufferedReader br = new BufferedReader(
							new InputStreamReader(
							s.getInputStream()));
				PrintWriter pw = new PrintWriter(
							s.getOutputStream());
				String login  = br.readLine();
				String passwd = br.readLine();

				if(login.equals("admin") && passwd.equals("admin"))
					pw.println("OK");
				else
					pw.println("NO");
				pw.flush();
				pw.close();
				br.close();
				s.close();
			} catch(Exception e) {}
		}
	}
}
```

ตัวอย่าง 3 : โปรแกรมที่ server ใช้เวลาทำงานนาน

- เขียนโปรแกรมให้ server โดยกำหนดให้งานที่ server ให้บริการต่อ client ใช้เวลานานมาก
  - Server ให้บริการที่ port หมายเลข 30000
  - Client ติดต่อไปยังส่งเลข 1 ค่า (วินาที) [มาจาก args[0]]
  - Server จะหลับรอเป็นจำนวนวินาทีที่ Client ส่งเข้ามา จากนั้นส่งคำว่า OK คืนให้กับ Client

โปรแกรม Client

```
import java.net.*;
import java.io.*;
public class Client3 {
      public static void main(String[] args) {
               try {
                        Socket s = new Socket("127.0.0.1", 30000);
                        BufferedReader br = new BufferedReader(
                                          new InputStreamReader(
                                          s.getInputStream()));
                        PrintWriter pw = new PrintWriter(s.getOutputStream());

                        pw.println(args[0]);
                        pw.flush();

                        String msg = br.readLine();
                        System.out.println(msg);

                        br.close();
                        pw.close();
                        s.close();
               } catch (Exception e) { e.printStackTrace(); }

      }
}
```

โปรแกรม Server

```
import java.io.*;
import java.net.*;

public class Server3 {
	public static void main(String[] args) {
		ServerSocket servSocket = null;
		try {
			servSocket = new ServerSocket(30000);
		} catch(Exception e) {e.printStackTrace();}

		while(true) {
			try {
				Socket s = servSocket.accept();

				BufferedReader br = new BufferedReader(
					new InputStreamReader(
					s.getInputStream()));
				PrintWriter pw = new PrintWriter(
					s.getOutputStream());

				String sleep  = br.readLine();
				long sleepTime = Long.parseLong(sleep);
				try {
					Thread.sleep(sleepTime*1000);
				} catch(Exception se) {}

				pw.println("OK");
				pw.flush();
				pw.close();
				br.close();
				s.close();
			} catch(Exception e) {}
		}
	}
}
```

ผลการรัน

- ถ้าสังเกตุผลการรันของโปรแกรมตัวอย่างที่ 3
  - จะเห็นว่า **Server** สามารถให้บริการ **Client** ได้ทีละ 1 ตัวเท่านั้น
  - **Server** จะให้บริการ **Client** ตัวถัดไปต่อเมื่อ **Server** ได้ให้บริการกับ **Client** เสร็จสิ้นก่อน
  - ซึ่งการขึ้นจากเมธอด **accept** ที่จะถูกเรียกก็ต่อเมื่อ **server** ทำงานต่างๆ ใน **loop while** เรียบร้อยก่อน
  - ปัญหาจึงทำให้ถ้า **server** ให้บริการบางอย่างที่ต้องใช้เวลาในการทำงานนานมาก **server** จะไม่สามารถให้บริการกับ **client** อื่นได้เลยในช่วงเวลานั้น
  - **วิธีแก้ไข** เขียนให้ **server** รองรับการทำงานแบบ **multi-thread**

การเขียน Server แบบ multi-thread

```java
import java.io.*;
import java.net.*;

public class Server extends Thread {
    Socket s;

    public Server(Socket s) {
        this.s = s;
    }

    public void run() {
        // CODE HERE
    }

    public static void main(String[] args) {
        ServerSocket server = null;
        try {
            server = new ServerSocket(2000);
        } catch (Exception e) {
            System.exit(1);
        }

        while(true) {
            try {
                Socket client = server.accept();
                Thread t = new Server(client);
                t.start();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

แก้ไขตัวอย่างที่ 3 ให้เป็นแบบ multi-thread

1

```
import java.io.*;
import java.net.*;

public class Server4 extends Thread {
      Socket s = null;

      public Server4(Socket s) {
              this.s = s;
      }

      public  void run() {
              try {
                      BufferedReader br = new BufferedReader(
                                          new InputStreamReader(
                                          s.getInputStream()));
                      PrintWriter pw = new PrintWriter(
                                       s.getOutputStream());

                      String sleep  = br.readLine();
                      long sleepTime = Long.parseLong(sleep);
                      try {
                              Thread.sleep(sleepTime*1000);
                      } catch(Exception se) {}

                      pw.println("OK");
                      pw.flush();
                      pw.close();
                      br.close();
                      s.close();
              } catch(Exception e) { e.printStackTrace(); }
      }
```

```
      public static void main(String[] args) {
              ServerSocket servSocket = null;
              try {
                      servSocket = new ServerSocket(30000);
              } catch(Exception e) {e.printStackTrace();}

              while(true) {
                      try {
                              Socket s = servSocket.accept();
                              Server4 serverThread = new Server4(s);
                              serverThread.start();
                      } catch(Exception e) {}
              }
      }
}
```

2

Code ของ client ไม่จำเป็นต้องมีการเปลี่ยนแปลงใดๆ ทั้งสิ้น

ปรับเปลี่ยนให้เป็น ThreadPool

```java
import java.util.concurrent.*;
import java.io.*;
import java.net.*;

public class ServerTP3 implements Runnable {
    Socket s = null;
    public ServerTP3(Socket s) {
        this.s = s;
    }

    public void run() {
        try {
            BufferedReader br = new BufferedReader(
                                new InputStreamReader(
                                s.getInputStream()));
            PrintWriter pw = new PrintWriter(s.getOutputStream());

            String sleep = br.readLine();
            long sleepTime = Long.parseLong(sleep);

            try {
                Thread.sleep(sleepTime * 1000);
            } catch(Exception se) {}

            pw.println("OK");
            pw.flush();
            pw.close(); br.close(); s.close();
        } catch(Exception e) {}

    }
```

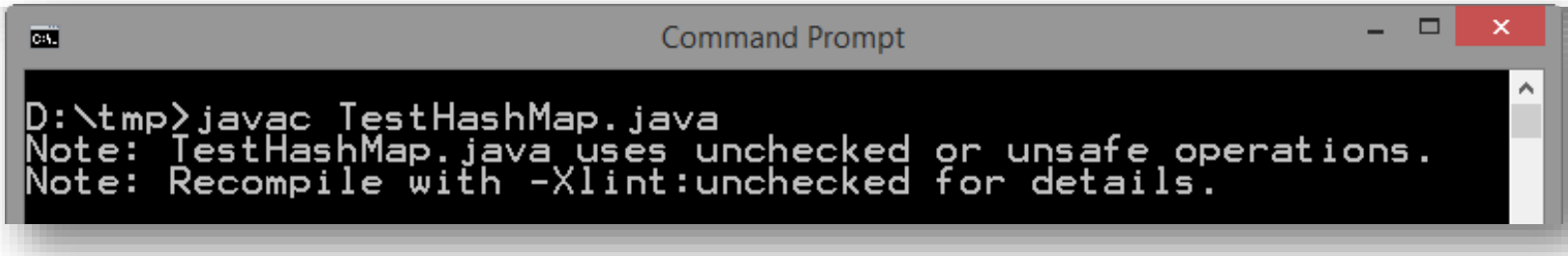
```java
    public static void main(String[] args) {
        ServerSocket serv = null;
        ExecutorService es = Executors.newFixedThreadPool(10);

        try {
            serv = new ServerSocket(30000);
        } catch(Exception e) {System.exit(1);}

        while(true) {
            try {
                Socket s = serv.accept();
                ServerTP3 st = new ServerTP3(s);
                es.execute(st);
            } catch(Exception e) {}
        }
    }
}
```

Java HashMap

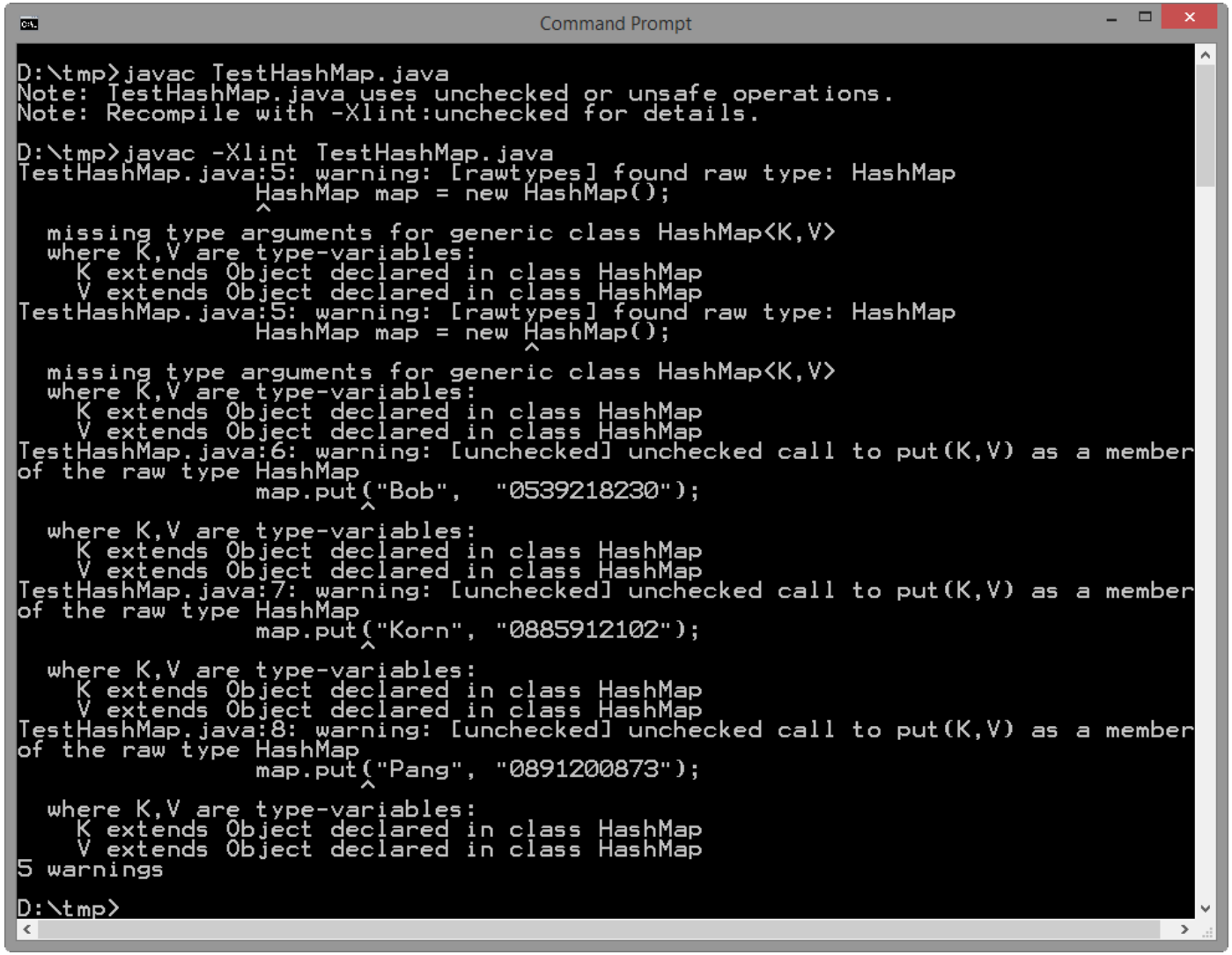
- HashMap อยู่ใน package java.util.*;
- HashMap มีการทำงานในลักษณะของ <key, value>
  - คือมี key เหมือนกับเป็น ID
  - โดยแต่ละ key จะมีค่า (value) ของ key นั้นๆ
  - hashMap จะมีการค้นหาข้อมูลที่รวดเร็ว O(1)
- เมธอดที่สำคัญของ Class HashMap มีอยู่ 2 เมธอดคือ
  - **put**(Object key, Object value)
    - เป็นการเก็บข้อมูล <key, value> ลงในตาราง hash
  - Object **get**(Object key)
    - เป็นการดึงค่า value ออกจากตาราง hash โดยใช้ key เป็นตัวค้นหา
    - คืนค่า null ถ้าไม่มี key ใน HashMap

ตัวอย่าง : การใช้งาน HashMap

```
import java.util.*;

public class TestHashMap {
    public static void main(String[] args) {
        HashMap map = new HashMap();
        map.put("Bob",  "0539218230");
        map.put("Korn", "0885912102");
        map.put("Pang", "0891200873");

        System.out.println("Pang :" + map.get("Pang"));
        System.out.println("Bob  :" + map.get("Bob"));
        System.out.println("Korn :" + map.get("Korn"));
    }
}
```



ตรวจสอบปัญหาของการใช้งาน HashMap



แก้ปัญหาต้องระบุประเภทของข้อมูล

```
import java.util.*;

public class TestHashMap {
    public static void main(String[] args) {
        HashMap<String, String> map = new HashMap<String, String>();
        map.put("Bob",  "0539218230");
        map.put("Korn", "0885912102");
        map.put("Pang", "0891200873");

        System.out.println("Pang :" + map.get("Pang"));
        System.out.println("Bob  :" + map.get("Bob"));
        System.out.println("Korn :" + map.get("Korn"));
    }
}
```