Exercise 1: Extending Thread class

In this exercise, you are going to learn how to create and start a thread execution by writing a class that extends Thread class. You will learn how to start the thread by either not having the start() method in the constructor of the subclass or having it in the constructor of the subclass.

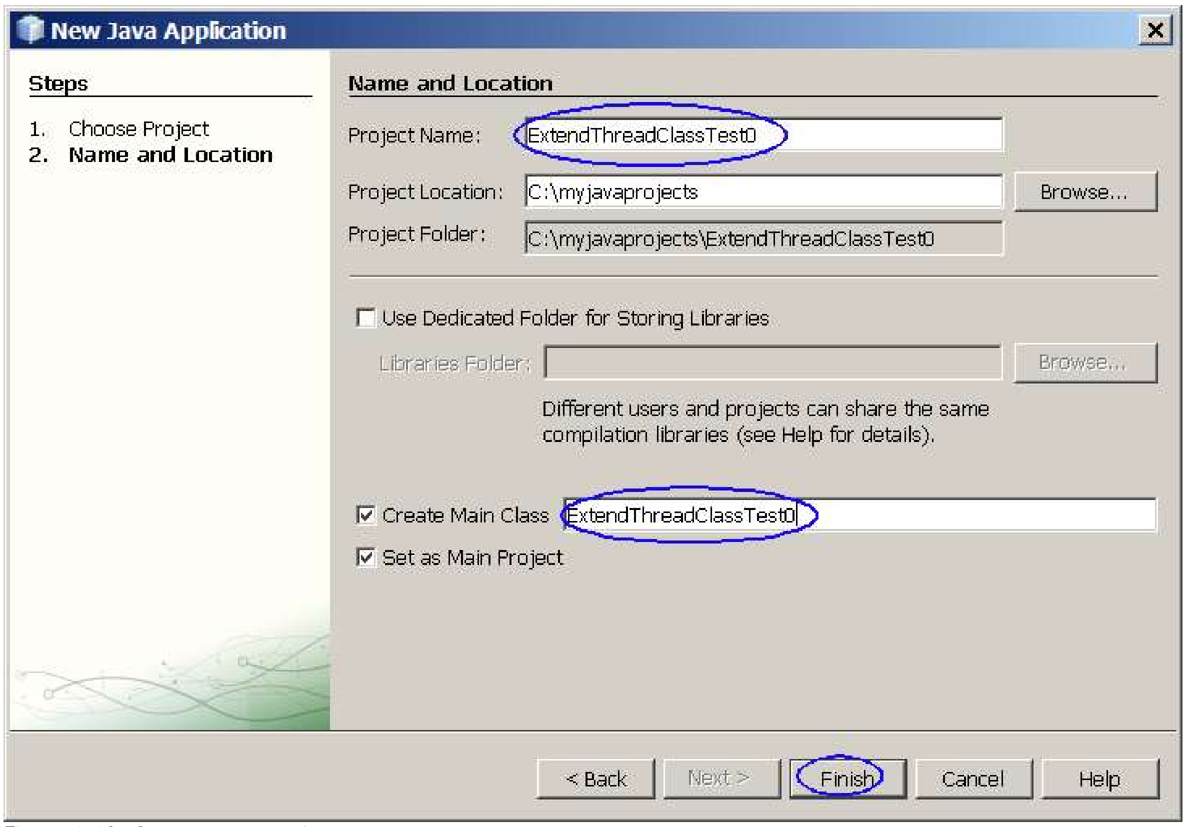
1. **The start() method is NOT in the constructor of the subclass**
2. **The start() method is in the constructor of the subclass**

(1.1) The start() method is NOT in the constructor of the subclass

0. Start NetBeans IDE if you have not done so yet.
1. Create a new NetBeans project

- Select **File**->**New Project (Ctrl+Shift+N)**. The **New Project** dialog box appears.
- Under **Choose Project** pane, select **Java** under **Categories** and **Java Application** under **Projects**.
- Click **Next**.

- Under **Name and Location** pane, for the **Project Name** field, type in **ExtendThreadClassTest0** as project name.
- For **Create Main Class** field, type in **ExtendThreadClassTest0**. (Figure-1.10 below)
- Click **Finish**.


New Java Application

Steps
1. Choose Project
2. Name and Location

Name and Location
Project Name: ExtendThreadClassTest0
Project Location: C:\myjavaprojects
Project Folder: C:\myjavaprojects\ExtendThreadClassTest0

Use Dedicated Folder for Storing Libraries
Libraries Folder:
Different users and projects can share the same compilation libraries (see Help for details).

[✓] Create Main Class ExtendThreadClassTest0
[✓] Set as Main Project

Figure-1.10: Create a new project

- Observe that **ExtendThreadClassTest0** project appears and IDE generated **ExtendThreadClassTest0.java** is displayed in the source editor window of NetBeans IDE.

2. Modify the IDE generated **ExtendThreadClassTest0.java** as shown in Code-1.11 below. Study the code by paying special attention to the bold fonted parts. Note that the **start()** is invoked after the object instance of **PrintNameThread** class is created.

```
public class ExtendThreadClassTest0 {

    public static void main(String args[]) {

        // Create object instance of a class that is subclass of Thread class
        System.out.println("Creating PrintNameThread object instance..");
        PrintNameThread pnt1 =
               new PrintNameThread("A");

        // Start the thread by invoking start() method
        System.out.println("Calling start() method of " + pnt1.getName() + " thread");
        pnt1.start();

    }
}
```
Code-1.11: ExtendThreadClassTest0.java

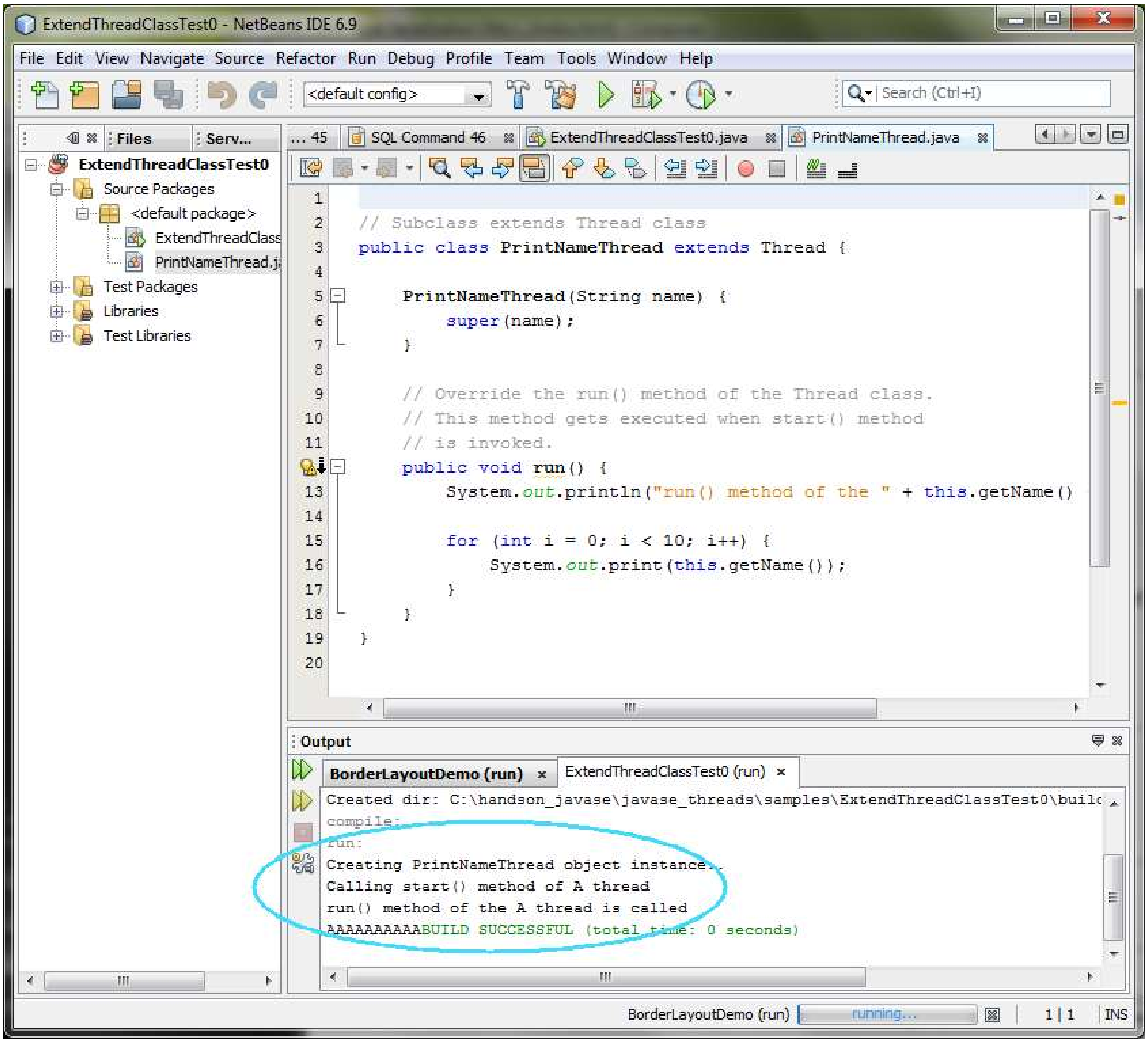
3. Write **PrintNameThread.java** as shown in Code-1.12 below.

```
// Subclass extends Thread class
public class PrintNameThread extends Thread {

   PrintNameThread(String name) {
      super(name);
   }

   // Override the run() method of the Thread class.
   // This method gets executed when start() method
   // is invoked.
   public void run() {
      System.out.println("run() method of the " + this.getName() + " thread is called" );

      for (int i = 0; i < 10; i++) {
         System.out.print(this.getName());
      }
   }
}
```
Code-1.12: PrintNameThread.java

4. Build and run the project

- Right click **ExtendThreadClassTest0** project and select **Run**.
- Observe the result in the **Output** window. (Figure-1.13 below)

```
Creating PrintNameThread object instance..
Calling start() method of A thread
run() method of the A thread is called
AAAAAAAAAA
```
Figure-1.13: Result of running ExtendThreadClassTest0 application



```

// Subclass extends Thread class
public class PrintNameThread extends Thread {

    PrintNameThread(String name) {
        super(name);
    }

    // Override the run() method of the Thread class.
    // This method gets executed when start() method
    // is invoked.
    public void run() {
        System.out.println("run() method of the " + this.getName()

        for (int i = 0; i < 10; i++) {
            System.out.print(this.getName());
        }
    }
}

```

```
Created dir: C:\handson_javase\javase_threads\samples\ExtendThreadClassTest0\builc
compile:
run:
Creating PrintNameThread object instance..
Calling start() method of A thread
run() method of the A thread is called
AAAAAAAAAABUILD SUCCESSFUL (total time: 0 seconds)
```

5. Modify the **ExtendThreadClassTest0.java** as shown in Code-1.15 below. The code fragments that need to be added are highlighted

in **bold and blue-colored** font.

```
public class ExtendThreadClassTest0 {

    public static void main(String args[]) {

         // Create object instance of a class that is subclass of Thread class
        System.out.println("Creating PrintNameThread object instance..");
        PrintNameThread pnt1 =
                new PrintNameThread("A");

        // Start the thread by invoking start() method
        System.out.println("Calling start() method of " + pnt1.getName() + " thread");
        pnt1.start();

        System.out.println("Creating PrintNameThread object instance..");
        PrintNameThread pnt2 =
                new PrintNameThread("B");
        System.out.println("Calling start() method of " + pnt2.getName() + " thread");
        pnt2.start();

        System.out.println("Creating PrintNameThread object instance..");
        PrintNameThread pnt3 =
                new PrintNameThread("C");
        System.out.println("Calling start() method of " + pnt3.getName() + " thread");
        pnt3.start();
    }
}
```

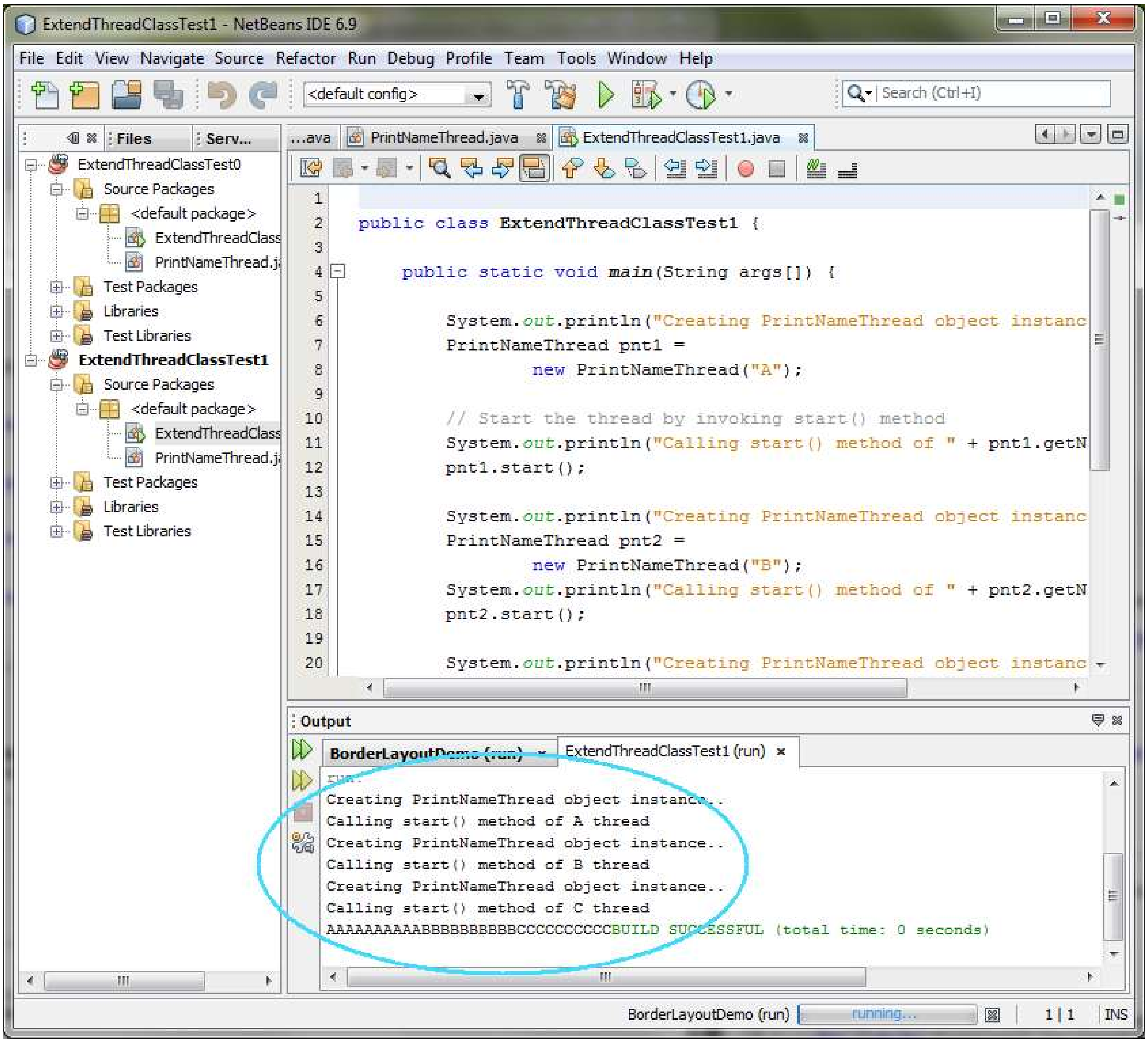
Code-1.15: Modified ExtendThreadClassTest0.java

6. Build and run the project

- Right click **ExtendThreadClassTest0** project and select **Run**.
- Observe the result in the **Output** window. (Figure-1.16 below)

```
Creating PrintNameThread object instance..
Calling start() method of A thread
Creating PrintNameThread object instance..
Calling start() method of B thread
AAAAAAAAAACreating PrintNameThread object instance..
BCalling start() method of C thread
BBBBBBBBBCCCCCCCCCC
```

Figure-1.16: Result



```
public class ExtendThreadClassTest1 {

    public static void main(String args[]) {

        System.out.println("Creating PrintNameThread object instanc
        PrintNameThread pnt1 =
                new PrintNameThread("A");

        // Start the thread by invoking start() method
        System.out.println("Calling start() method of " + pnt1.getN
        pnt1.start();

        System.out.println("Creating PrintNameThread object instanc
        PrintNameThread pnt2 =
                new PrintNameThread("B");
        System.out.println("Calling start() method of " + pnt2.getN
        pnt2.start();

        System.out.println("Creating PrintNameThread object instanc
```

```
run:
Creating PrintNameThread object instance..
Calling start() method of A thread
Creating PrintNameThread object instance..
Calling start() method of B thread
Creating PrintNameThread object instance..
Calling start() method of C thread
AAAAAAAAAABBBBBBBBBBCCCCCCCCCCBUILD SUCCESSFUL (total time: 0 seconds)
```

7. For your own exercise, modify **ExtendThreadClassTest0.java** as following. Build and run the application.

- Create and start another thread.
- Set the name of the thread as "MyOwn"

return to top of the exercise

(1.2) The start() method is in the constructor of the subclass

1. Create a new NetBeans project

- Select **File**->**New Project (Ctrl+Shift+N)**. The **New Project** dialog box appears.
- Under **Choose Project** pane, select **Java** under **Categories** and **Java Application** under **Projects**.
- Click **Next**.

- Under **Name and Location** pane, for the **Project Name** field, type in **ExtendThreadClassTest2** as project name.
- For **Create Main Class** field, type in **ExtendThreadClassTest2**.
- Click **Finish**.

- Observe that **ExtendThreadClassTest2** project appears and IDE generated **ExtendThreadClassTest2.java** is displayed in the source editor window of NetBeans IDE.

2. Modify the IDE generated **ExtendThreadClassTest2.java** as shown in Code-1.21 below.

```
public class ExtendThreadClassTest2 {

   public static void main(String args[]) {


      PrintNameThread pnt1 =
            new PrintNameThread("A");
```

```
    PrintNameThread pnt2 =
        new PrintNameThread("B");


    PrintNameThread pnt3 =
        new PrintNameThread("C");

  }
}
```
Code-1.21: ExtendThreadClassTest2.java

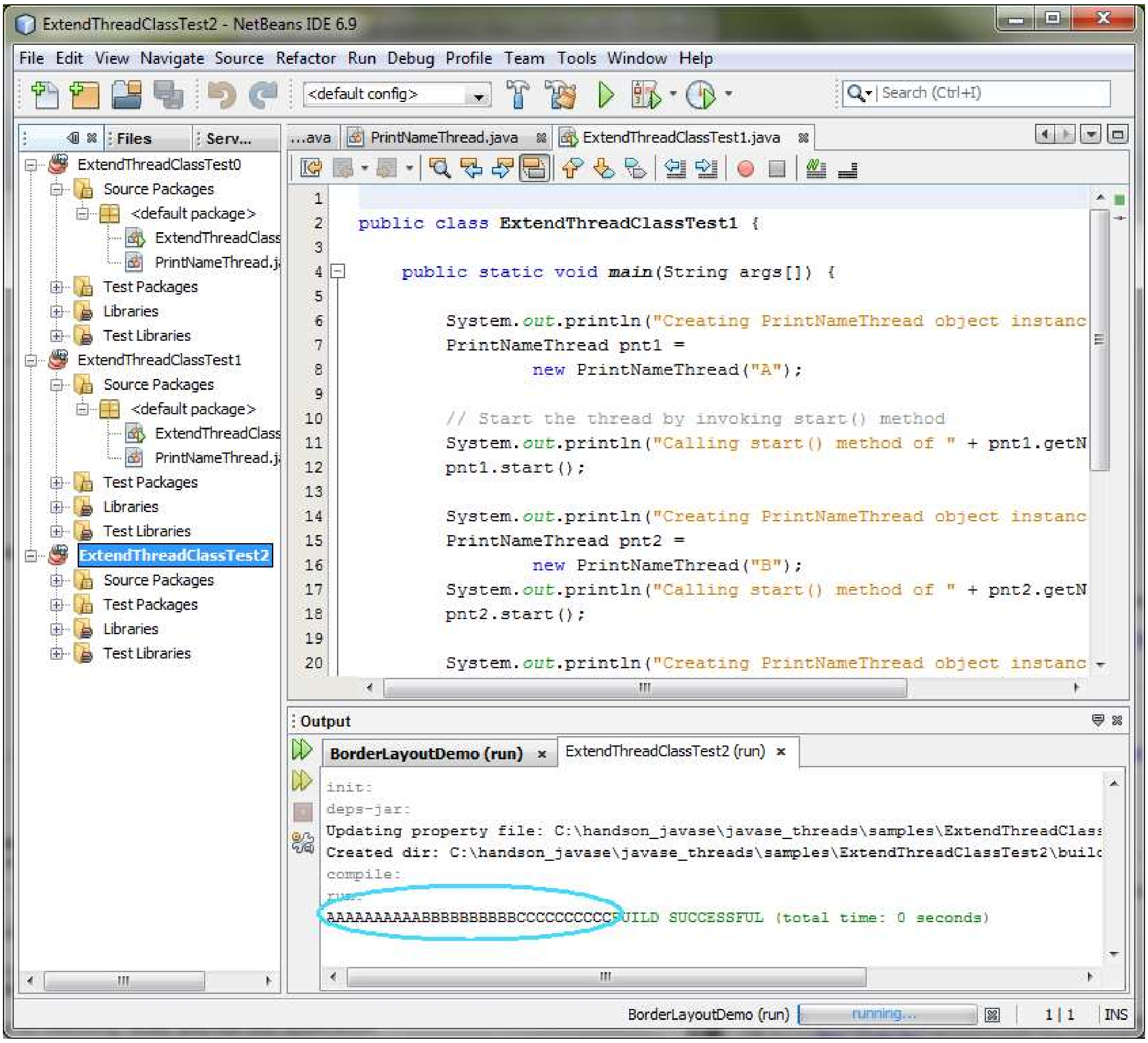
3. Write **PrintNameThread.java** as shown in Code-1.22 below. Note that the **start**() method is invoked as part of the constructor method of the **PrintNameThread** class.

```
public class PrintNameThread extends Thread {

  PrintNameThread(String name) {
    super(name);
    // start() method is inside the constructor of the subclass
    start();
  }

  public void run() {
    String name = getName();
    for (int i = 0; i < 10; i++) {
      System.out.print(name);
    }
  }
}
```
Code-1.22: PrintNameThread.java

4. Build and run the project

- Right click **ExtendThreadClassTest2** project and select **Run**.
- Observe the result in the **Output** window. (Figure-1.23 below)

```
AAAAAAAAAABBBBBBBBBBCCCCCCCCCC
```
Figure-1.23: Result of running ExtendThreadClassTest2 application



```
public class ExtendThreadClassTest1 {

    public static void main(String args[]) {

        System.out.println("Creating PrintNameThread object instanc
        PrintNameThread pnt1 =
                new PrintNameThread("A");

        // Start the thread by invoking start() method
        System.out.println("Calling start() method of " + pnt1.getN
        pnt1.start();

        System.out.println("Creating PrintNameThread object instanc
        PrintNameThread pnt2 =
                new PrintNameThread("B");
        System.out.println("Calling start() method of " + pnt2.getN
        pnt2.start();

        System.out.println("Creating PrintNameThread object instanc
```

```
init:
deps-jar:
Updating property file: C:\handson_javase\javase_threads\samples\ExtendThreadClas
Created dir: C:\handson_javase\javase_threads\samples\ExtendThreadClassTest2\build
compile:
run:
AAAAAAAAAABBBBBBBBBBCCCCCCCCCCBUILD SUCCESSFUL (total time: 0 seconds)
```

5. For your own exercise, modify **ExtendThreadClassTest2.java** as following. Build and run the application.

- Create and start another thread.
- Set the name of the thread as "MyOwn"

return to top of the exercise

Summary

In this exercise, you have learned how to create and start a thread by extending Thread class.

return to the top

Exercise 2: Implement Runnable interface

In this exercise, you are going to create and start a thread by writing a class that implements Runnable interface.

1. **Create and start a thread by implementing Runnable interface - start() method is not in the constructor**
2. **Create and start a thread by implementing Runnable interface - start() method is in the constructor**

(2.1) Create and start a thread by implementing Runnable interface - start() method is not in the constructor

1. Create a new NetBeans project

- Select **File**->**New Project (Ctrl+Shift+N)**. The **New Project** dialog box appears.
- Under **Choose Project** pane, select **Java** under **Categories** and **Java Application** under **Projects**.
- Click **Next**.

- Under **Name and Location** pane, for the **Project Name** field, type in **RunnableThreadTest1** as project name.

- For **Create Main Class** field, type in **RunnableThreadTest1**.
- Click **Finish**.

- Observe that **RunnableThreadTest1** project appears and IDE generated **RunnableThreadTest1.java** is displayed in the source editor window of NetBeans IDE.

2. Modify the IDE generated **RunnableThreadTest1.java** as shown in Code-2.11 below. Study the code by paying special attention to the bold fonted parts. Note that the **start()** method needs to be invoked explicitly after an object instance of the **PrintNameRunnable** class is created.

```
public class RunnableThreadTest1 {

    public static void main(String args[]) {

        PrintNameRunnable pnt1 = new PrintNameRunnable("A");
        Thread t1 = new Thread(pnt1);
        t1.start();

        PrintNameRunnable pnt2 = new PrintNameRunnable("B");
        Thread t2 = new Thread(pnt2);
        t2.start();

        PrintNameRunnable pnt3 = new PrintNameRunnable("C");
        Thread t3 = new Thread(pnt3);
        t3.start();

    }
}
```
Code-2.11: RunnableThreadTest1.java

3. Write **PrintNameRunnable.java** as shown in Code-2.12 below.

```
// The class implements Runnable interface
class PrintNameRunnable implements Runnable {

    String name;

    PrintNameRunnable(String name) {
        this.name = name;
    }

    // Implementation of the run() defined in the
    // Runnable interface.
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.print(name);
        }
    }
}
```
Code-2.12: PrintNameRunnable.java

4. Build and run the project

- Right click **RunnableThreadTest1** project and select **Run**.
- Observe the result in the **Output** window. (Figure-2.13 below)

```
ACBACBACBACBACABCABCABCABCABCB
```
Figure-2.13: Result of running RunnableThreadTest1 application

5. For your own exercise, do the following. Build and run the application.

- Create another class called **MyOwnRunnableClass** that implements Runnable interface
- **MyOwnRunnableClass** displays values 1 to 10 inside its run() method
- Modify **RunnableThreadTest1.java** to start 2 thread instances of **MyOwnRunnableClass**.

return to top of the exercise

(2.2) Create and start a thread by implementing Runnable interface - start() method is in the constructor

1. Create a new NetBeans project

- Select **File**->**New Project (Ctrl+Shift+N)**. The **New Project** dialog box appears.
- Under **Choose Project** pane, select **Java** under **Categories** and **Java Application** under **Projects**.
- Click **Next**.

- Under **Name and Location** pane, for the **Project Name** field, type in **RunnableThreadTest2** as project name.
- For **Create Main Class** field, type in **RunnableThreadTest2**.
- Click **Finish**.

- Observe that **RunnableThreadTest2** project appears and IDE generated **RunnableThreadTest2.java** is displayed in the source editor window of NetBeans IDE.

2. Modify the IDE generated **RunnableThreadTest2.java** as shown in Code-2.21 below. Study the code by paying special attention to the bold fonted parts.

```
public class RunnableThreadTest2 {

   public static void main(String args[]) {

      // Since the constructor of the PrintNameRunnable
      // object creates a Thread object and starts it,
      // there is no need to do it here.
      new PrintNameRunnable("A");

      new PrintNameRunnable("B");
      new PrintNameRunnable("C");
   }
}
```
Code-2.21: RunnableThreadTest2.java

3. Write PrintNameRunnable.java as shown in Code-2.22 below. Study the code by paying special attention to the bold fonted parts. Note that the start() method is in the constructor of the PrintNameRunnable class.

```
// The class implements Runnable interface
class PrintNameRunnable implements Runnable {

   Thread thread;

   PrintNameRunnable(String name) {
      thread = new Thread(this, name);
      thread.start();
   }

   // Implementation of the run() defined in the
   // Runnable interface.
   public void run() {
      String name = thread.getName();
      for (int i = 0; i < 10; i++) {
         System.out.print(name);
      }
   }
}
```
Code-2.22: PrintNameRunnable.java

4. Build and run the project

- Right click **RunnableThreadTest2** project and select **Run**.
- Observe the result in the **Output** window. (Figure-1.23 below)

```
ABCABCABCABCABCABCABCBACBACBAC
```
Figure-2.23: Result of running RunnableThreadTest2 application

return to top of the exercise

Summary

In this exercise, you have learned how to create a class that implements Runnable interface and starts a thread.