COMP 249: Object Oriented Programming II

Tutorial 4:

Inheritance and Polymorphism

Output Question 1

- **What is the output of the following program?**

```
class Base {
        public void print() {
                System.out.println("Base");
        }
}
class Derived extends Base {
        public void print() {
                System.out.println("Derived");
        }
}
class Main{
        public static void doPrint( Base o ) {
                o.print();
        }
        public static void main(String[] args) {
                Base x = new Base();
                Base y = new Derived();
                Derived z = new Derived();
                doPrint(x);
                doPrint(y);
                doPrint(z);
        }
}
```

Output Question 2

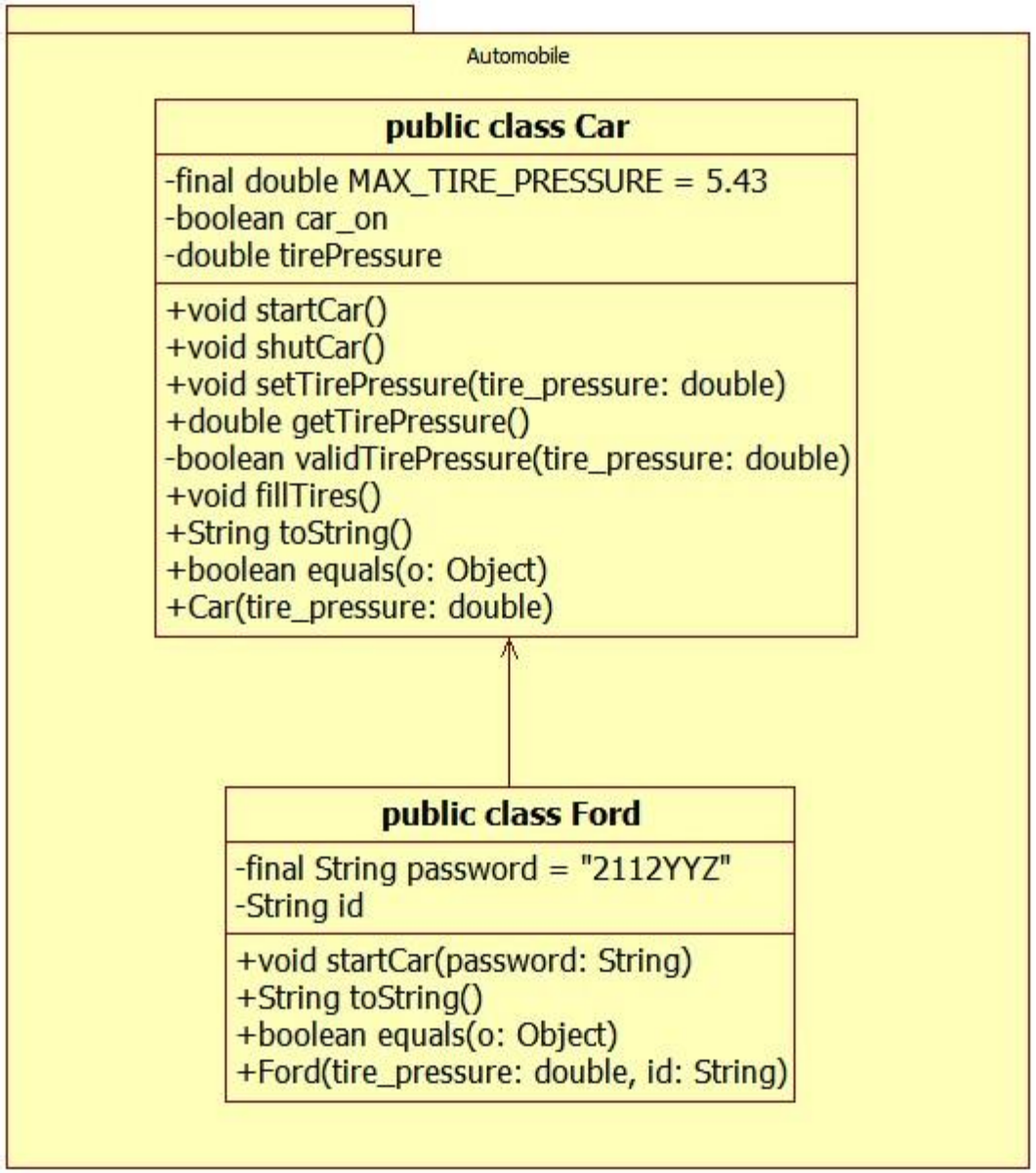
- What are the errors in the following program and how they can be fixed ?

```
public class A {
    private int a = 100;
    public void setA( int value) {
        a = value;
}   public int getA() {
        return a;
    }
}
```

```
public class OOPExercises {
    public static void main(String[] args)      {
            A objA = new A();
            System.out.println("in main(): ");
            System.out.println("objA.a = "+objA.a);
            objA.a = 222;
    }
}
```

Programming Question

- Implement the UML diagram on the right.
- Both Classes in package “Automobile”, Driver.java should be in a separate (default) package.
- *Privacy Legend:*
  - *+ means public*
  - *~ means package*
  - *# means protected*
  - *- means private*



Programming Question

- startCar(…) of Ford class should overload that of the Car class, which allows the user to optionally enter a password that would unlock special features not implemented in this program.
- Print an appropriate message if the password is correct.
- Method should call the startCar() of the superclass regardless of password validity.
- Overriding toString() from the Ford class should use that of its superclass.
- Proper implementation of the equals method consists of handling null references and object-attribute comparison.

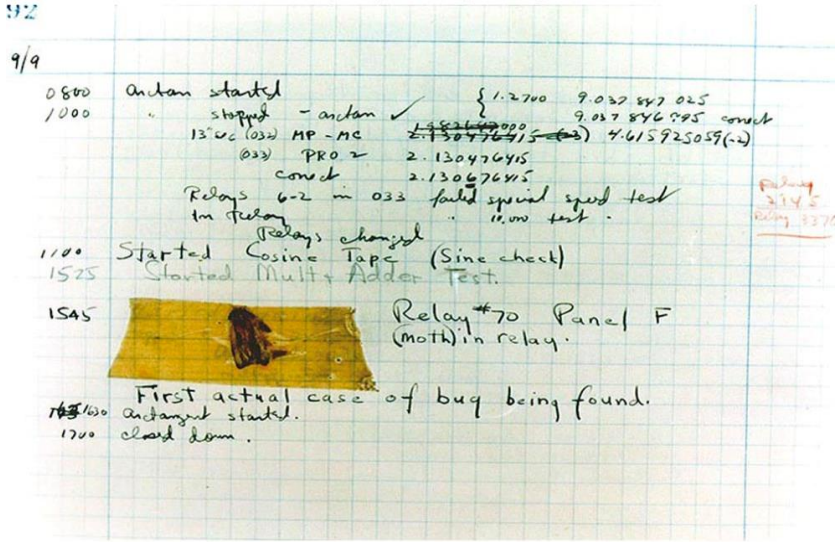
What is a software Bug ?

- A defect in a computer program causing it to malfunction
- Reasons:
  - Insufficient logic or erroneous logic
  - Design flaws
  - Hardware failures
  - Etc.

What is a software Bug ? Cont'd.

- First computer bug was in fact an actual Bug ☺
- 1945 at Harvard, a Moth trapped between two electrical relays of the Mark II Aiken Relay Calculator caused the whole machine to shut down.

Photo # NH 96566-KN (Color) First Computer "Bug", 1947



Some famous Bugs

- The European Space Agency's Ariane 5 Flight 501 was destroyed 40 seconds after takeoff (June 4, 1996) due to a bug in the on-board guidance software.
- NASA's Spirit rover became unresponsive on January 21, 2004, a few weeks after landing on Mars due to accumulation of too many files in the rover's flash memory.
- The 2003 North America blackout was triggered by a local outage that went undetected due to a race condition in General Electric Energy's XA/21 monitoring software.
- Smart ship USS Yorktown was left dead in the water in 1997 for nearly 3 hours after a divide by zero error.

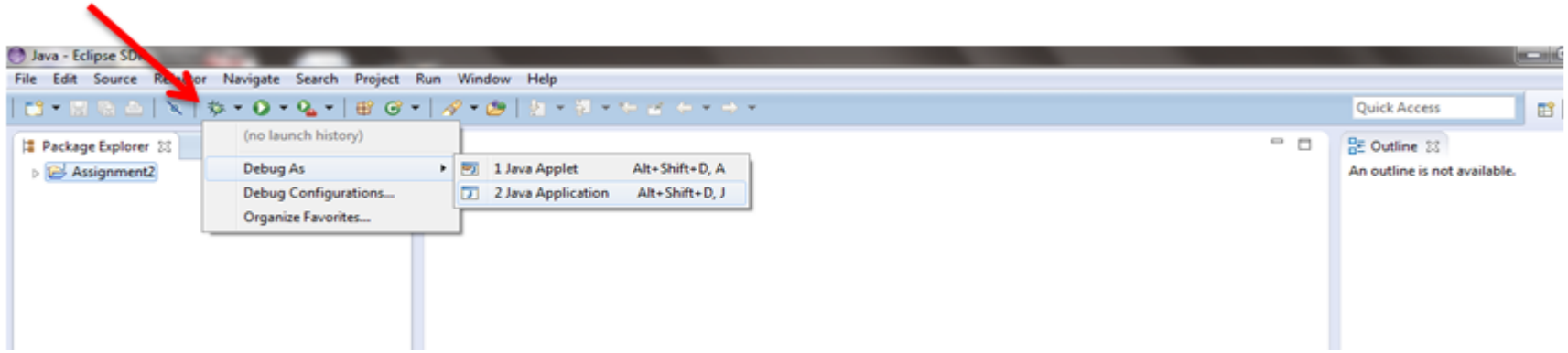
What is software Debugging?

- The process of investigating and fixing defects / bugs within a computer program.

- Methods of debugging
  - **printf** debugging
  - Log scraping
  - Post-mortem debugging
    - Debugging of the program after it has already crashed.
    - Create core dumps/crash dumps
    - Analyse using various tools : WinDbg , gdb
  - built-in debugging features in IDE Platforms, Visual Studio.NET, Eclipse, NetBeans etc.

Debug using Eclipse

- Debugging support on Eclipse
  - Provide and execution mode ‘Debug’
  - Lots of features to investigate the program behaviour while it’s executing.



Eclipse Debug view

Debug - BankApplication/src/Bank.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java | Debug

Debug

<terminated>Bank [Java Application]
<disconnected>Bank at localhost:5891
<terminated, exit value: 0>C:\Program Files (x86)\Java\jre7\bin\javaw.exe (2013-01-17 1:45:27 AM)

Expressions | Variables | Breakpoints

Name	Value
"compareContacts...ole - 1], item)"	
Add new expression	

Student.java | Driver.java | Bank.java

```
    there is not enough balance for withdraw
    do nothing */
     if (withdrawAmount >= 0 && withdrawAmount <= balance) {
         balance-= withdrawAmount;
     }
 }

 /** return balance */
 public double getbalance() {
     return balance;
 }

 /** return name */
 public String getName() {
     return name;
 }
```

Console | Tasks

<terminated> Bank [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (2013-01-17 1:45:27 AM)

```
Name: Gina Lollobrigida
balance: 5000.0
Interest Rate : 0.0
```

Writable | Smart Insert | 107 : 5

Eclipse Debug Features: Breakpoints

- Breakpoints
  - Set a particular breakpoint inside the code.
  - When executed in Debug mode, the execution get suspended at the breakpoint.
  - Gives the facility to investigate current execution status of the program.

Setting a breakpoint in Eclipse

efactor Navigate Search Project Run Window Help

Student.java Driver.java Bank.java

```
public BankAccount(String name) {
    this(name,0);
}

/** deposit money into the account */
public void deposit(double depositAmount) {
    if (depositAmount >= 0) {  // depositAmount must be positive
        balance+= depositAmount;
    }
}

/** withdraw money from the account */
public void withdraw(double withdrawAmount) {
/* If the withdrawAmount is negative or
```

Toggle Breakpoint Ctrl+Shift+B
Toggle Breakpoint Enablement
Go to Annotation Ctrl+1
Add Bookmark...
Add Task...
Show Quick Diff Ctrl+Shift+Q
Show Line Numbers
Folding
Preferences...
Breakpoint Properties...

```
Name: Gina Lollobrigida
balance: 5000.0
Interest Rate : 0.0
```

Eclipse Debug Features: Breakpoints cont'd.

Debug - BankApplication/src/Bank.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java | Debug

Debug

- Bank [Java Application]
  - Bank at localhost:5899
    - Thread [main] (Suspended (breakpoint at line 98 in BankAccount))
      - SavingsAccount(BankAccount).withdraw(double) line: 98
      - Bank.main(String[]) line: 123
    - C:\Program Files (x86)\Java\jre7\bin\javaw.exe (2013-01-17 1:50:41 AM)

Expressions | Variables | Breakpoints

Name	Value
"compareContacts...ole - 1], item)"	<error
Add new expression	

Student.java | Driver.java | Bank.java

```
        if (depositAmount >= 0) {  // depositAmount must be positive
            balance+= depositAmount;
        }
    }

    /** withdraw money from the account */
    public void withdraw(double withdrawAmount) {
    /* If the withdrawAmount is negative or
       there is not enough balance for withdraw
       do nothing */
        if (withdrawAmount >= 0 && withdrawAmount <= balance) {
            balance-= withdrawAmount;
        }
    }

    /** return balance */
    public double getbalance() {
```

Eclipse Debug Features: Different Views

- Execution View

Student.java | Driver.java | Bank.java

```
        if (depositAmount >= 0) {  // depositAmount must be positive
            balance+= depositAmount;
        }
    }

    /** withdraw money from the account */
    public void withdraw(double withdrawAmount) {
    /* If the withdrawAmount is negative or
       there is not enough balance for withdraw
       do nothing */
        if (withdrawAmount >= 0 && withdrawAmount <= balance) {
            balance-= withdrawAmou
        }
    }

    /** return balance */
    public double getbalance() {
```

withdrawAmount= 100.0

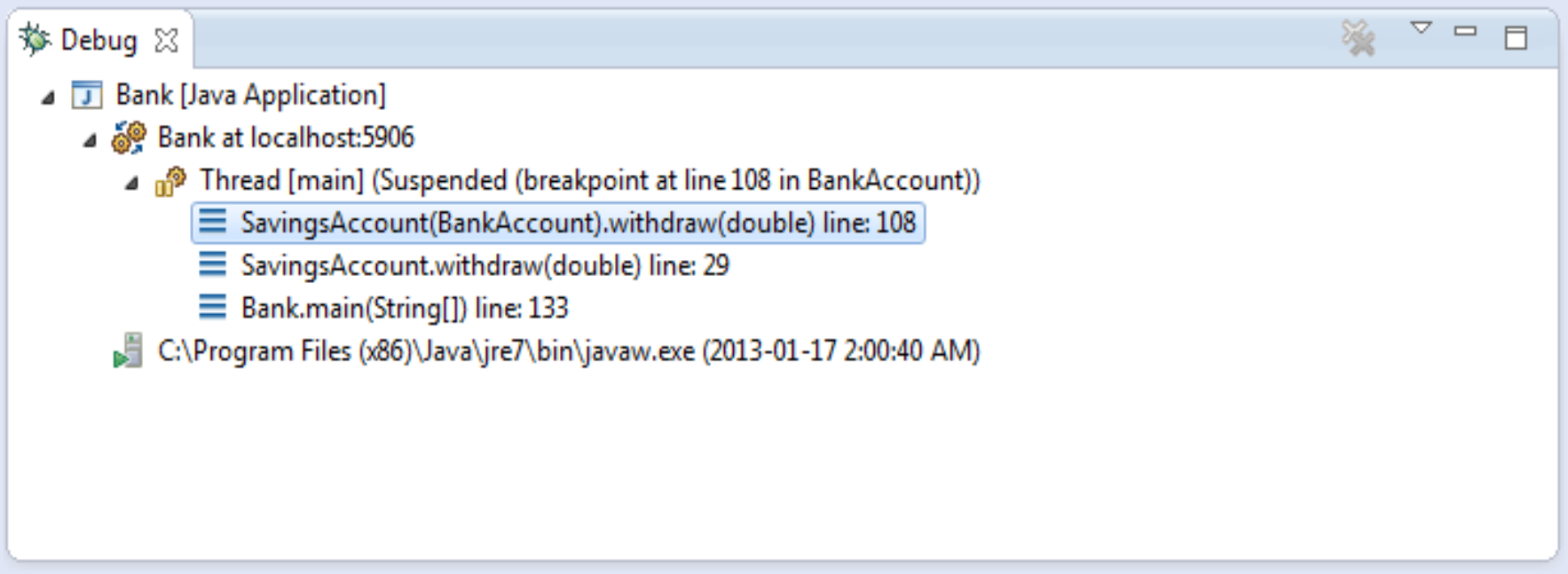
100.0

Console | Tasks

Bank [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (2013-01-17 1:50:41 AM)

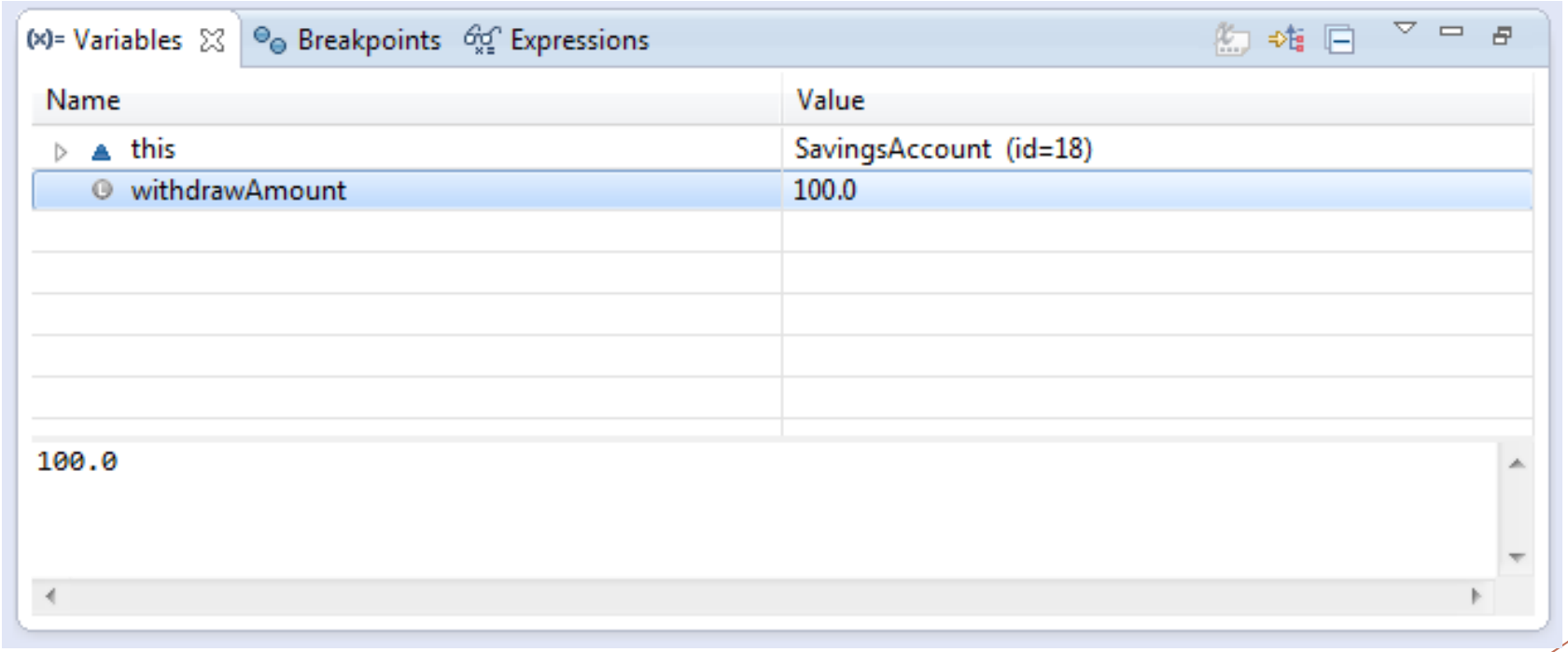
Eclipse Debug Features: Different Views cont'd.

- Stack View



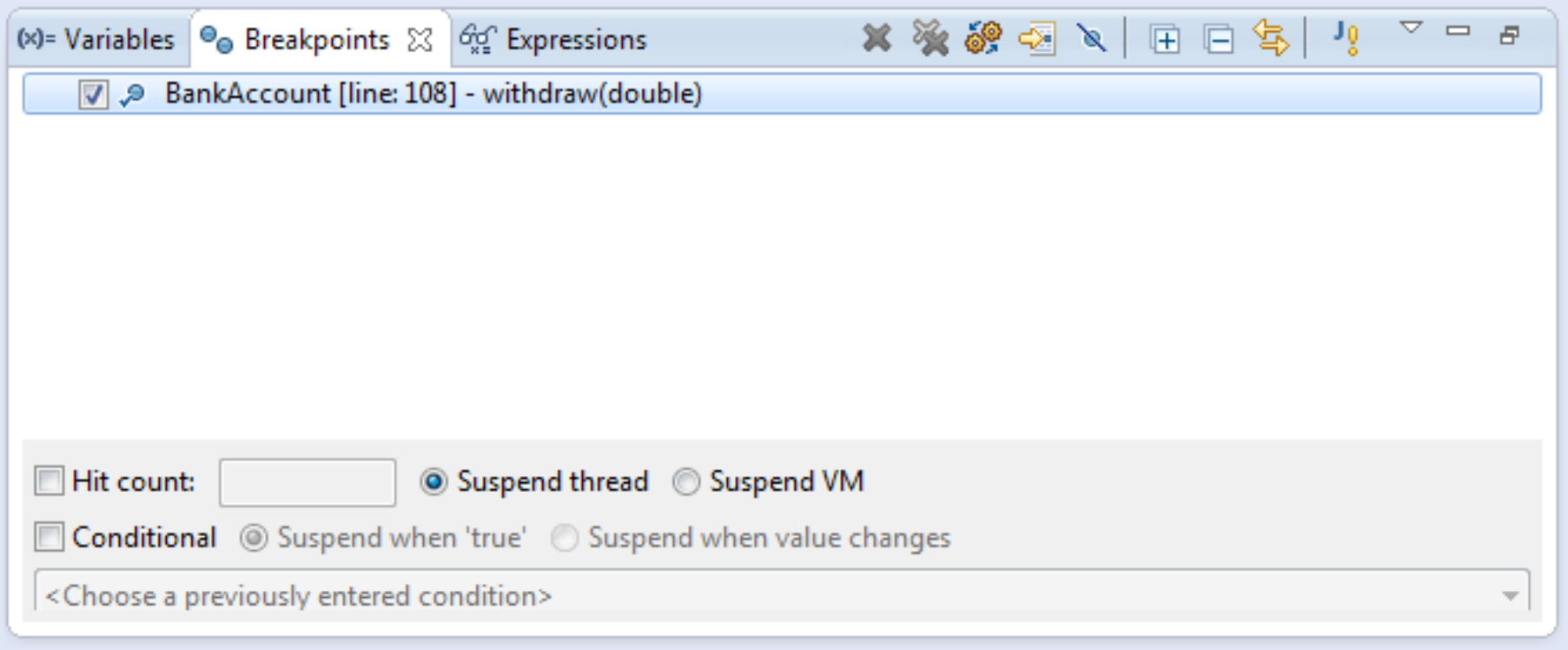
Eclipse Debug Features: Different Views cont'd.

- The Variables View



Eclipse Debug Features: Different Views cont'd.

- The Breakpoints View



Eclipse Debug Features: Line breakpoint properties

Debug - BankApplication/src/Bank.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java | Debug

Debug

- Bank [Java Application]
  - Bank at localhost:5906
    - Thread [main] (Suspended (breakpoint at line 108 in BankAccount))
      - SavingsAccount(BankAccount).withdraw(double) line: 108

Variables | Breakpoints | Expressions

BankAccount [line: 108] - withdraw(double)

Student.java | Driver.java | Bank.java

```
public void withdraw(double withdrawAmount) {
/* If the withdrawAmount is negative or
   there is not enough balance for withdraw
   do nothing */
    if (withdrawAmount >= 0 && withdrawAmount <= balance) {
        balance-= withdrawAmount
    }
}

/** return balance */
public double getbalance() {
    return balance;
}

/** return name */
public String getName() {
    return name;
}

public String toString() {
    return "Name: " + name + "\n
}
}
```

Properties for BankAccount [line:108] - withdraw(double)

type filter text

- Breakpoint Properties
- Filtering

Line Breakpoint

Type: BankAccount
Line Number: 108
Member: withdraw(double)

☑ Enabled
☐ Hit count: ◉ Suspend thread ◯ Suspend VM
☐ Conditional ◉ Suspend when 'true' ◯ Suspend when value changes

<Choose a previously entered condition>

OK | Cancel

Console | Tasks

Bank [Java Application] C:\Program Files (x86)\Java\jre

Eclipse Debug Features:
Line breakpoint properties cont'd.

- Set 'hit count' break points
  - The program suspends at the particular break point when it reaches the 'hit count' limit.

- Set 'conditional' break points
  - The program suspends at the particular break point when the given condition is true.

Eclipse Debug Features:
Line breakpoint properties cont'd.

Debug

- Bank [Java Application]
  - Bank at localhost:53981
    - Thread [main] (Suspended (breakpoint at line 116 in BankAccount))
      - SavingsAccount(BankAccount).withdraw(double) line: 116
      - SavingsAccount.withdraw(double) line: 31
      - Bank.main(String[]) line: 143
    - C:\Program Files (x86)\Java\jre7\bin\javaw.exe (2013-01-17 1:36:36 PM)

Variables | Breakpoints | Expressions

Name	Value
this	SavingsAccount (id=18)
withdrawAmount	40.0

40.0

Student.java | Driver.java | Bank.java

```
            balance+= depositAmount;
        }
    }

    /** withdraw money from the account */
    public void withdraw(double withdrawAmount) {
    /* If the withdrawAmount is negative or
       there is not enough balance for withdraw
       do nothing */
        if (withdrawAmount >= 0 && withdrawAmount <= bala
            balance-= withdrawAmount;
        }
    }

    /** return balance */
    public double getbalance() {
        return balance;
    }

    /** return name */
    public String getName() {
```

Properties for BankAccount [line:116] - withdraw(double)

type filter text

- Breakpoint Properties
- Filtering

Line Breakpoint

Type: BankAccount
Line Number: 116
Member: withdraw(double)

- [ ] Enabled
- [x] Hit count: 4 — (•) Suspend thread ( ) Suspend VM
- [ ] Conditional — (•) Suspend when 'true' ( ) Suspend when value changes

<Choose a previously entered condition>

OK | Cancel

Outline

- hasInterest : bo
- CheckingAcco
- CheckingAcco
- getMonthlyCh
- toString() : Stri
- BankAccount
- balance : doub
- name : String
- BankAccount(S
- BankAccount(S
- deposit(double
- withdraw(doub
- getbalance() : (
- getName() : St
- toString() : Stri
- Bank
- main(String[])

Console | Tasks

Bank [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (2013-01-1

Writable | Smart Insert | 116 : 1

Eclipse Debug Features: Conditional line breakpoints cont'd.

Debug - BankApplication/src/Bank.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java | D

Debug

- Bank [Java Application]
  - Bank at localhost:5920
    - Thread [main] (Suspended (breakpoint at line 108 in BankAccount))
      - SavingsAccount(BankAccount).withdraw(double) line: 108
      - SavingsAccount.withdraw(double) line: 29
      - Bank.main(String[]) line: 133
    - C:\Program Files (x86)\Java\jre7\bin\javaw.exe (2013-01-17 2:09:51 AM)

Variables | Breakpoints | Expressions

Name	Value
this	SavingsAccount (id=18)
withdrawAmount	101.0

101.0

Student.java | Driver.java | Bank.java

```
/** withdraw money from the account */
public void withdraw(double withdrawAmount) {
/* If the withdrawAmount is negative or
   there is not enough balance for withdraw
   do nothing */
    if (withdrawAmount >= 0 && withdrawAmount <= balance)
        balance-= withdrawAmount;
    }
}

/** return balance */
public double getbalance() {
    return balance;
}
```

Console | Tasks

Bank [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (2013-01-17 2:0

Properties for BankAccount [line:108] - withdraw(double)

type filter text

- Breakpoint Properties
- Filtering

Line Breakpoint

Type: BankAccount
Line Number: 108
Member: withdraw(double)

☑ Enabled
☐ Hit count: ◉ Suspend thread ○ Suspend VM
☑ Conditional ◉ Suspend when 'true' ○ Suspend when value changes

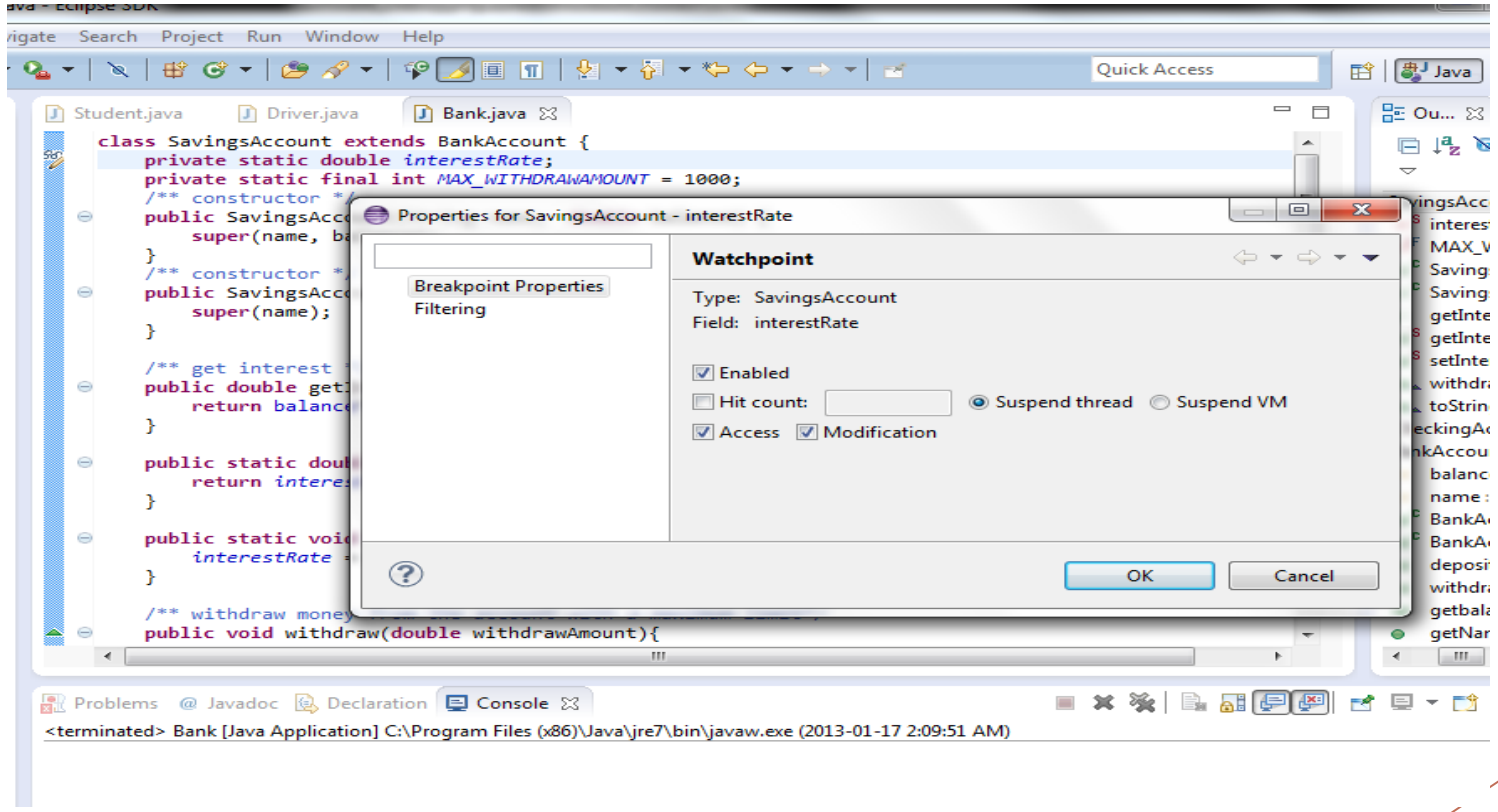
<Choose a previously entered condition>

```
withdrawAmount > 100
```

OK | Cancel

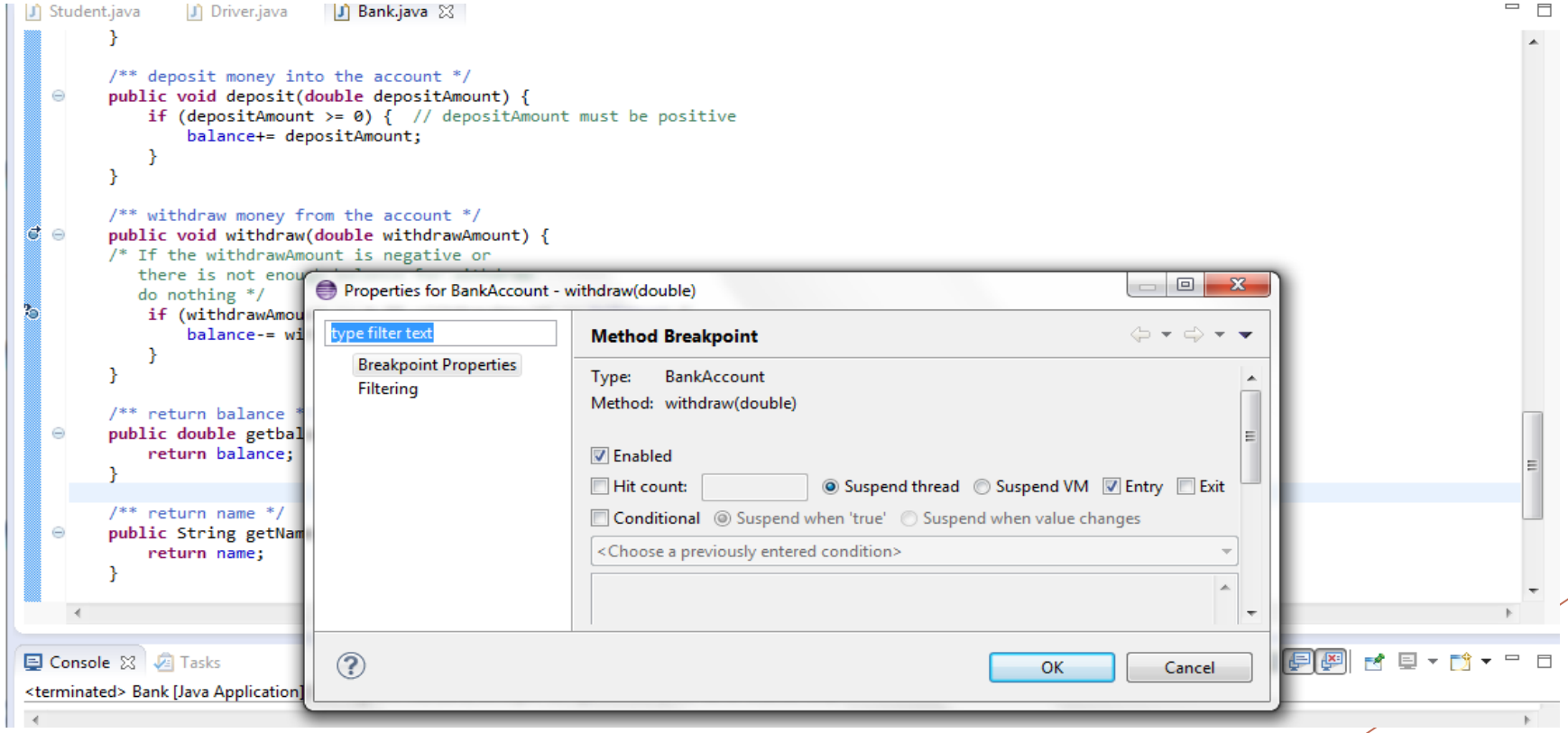
Eclipse Debug Features: watchpoints

- A watchpoint is a break point which targeted on a particular field.

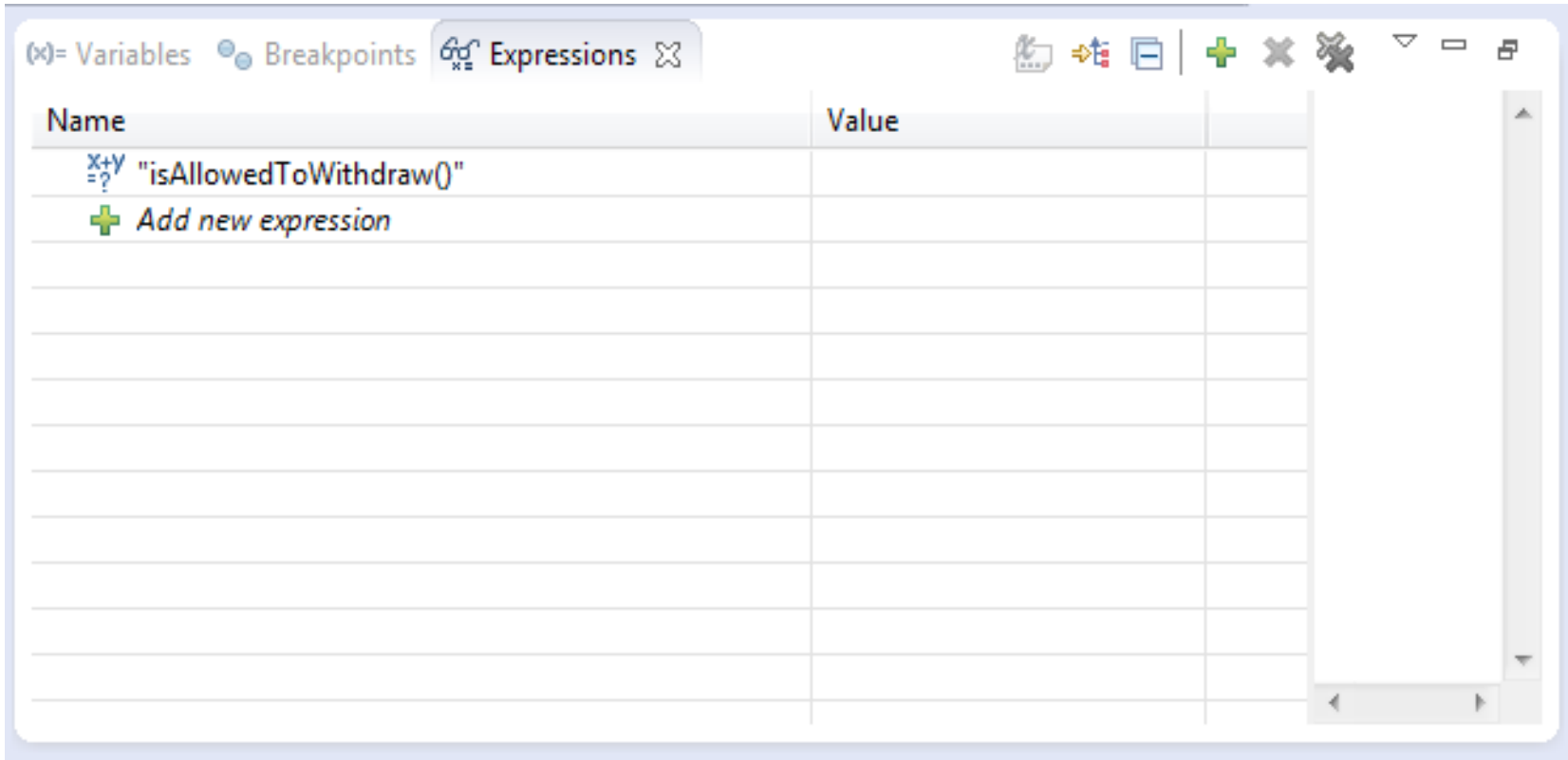


Eclipse Debug Features: Method breakpoints

- A method breakpoint is a break point which targeted on a particular method.



Eclipse Debug Features: Expression View & Watch Expressions



Eclipse Debug Features: Expression View & Watch Expressions cont'd.

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java | Deb

Debug

- Bank [Java Application]
  - Bank at localhost:5936
    - Thread [main] (Suspended (breakpoint at line 30 in SavingsAccount))
      - SavingsAccount.withdraw(double) line: 30
      - Bank.main(String[]) line: 141
  - C:\Program Files (x86)\Java\jre7\bin\javaw.exe (2013-01-17 2:32:18 AM)

Variables | Breakpoints | Expressions

Name	Value
"isAllowedToWithdraw()"	true
Add new expression	

true

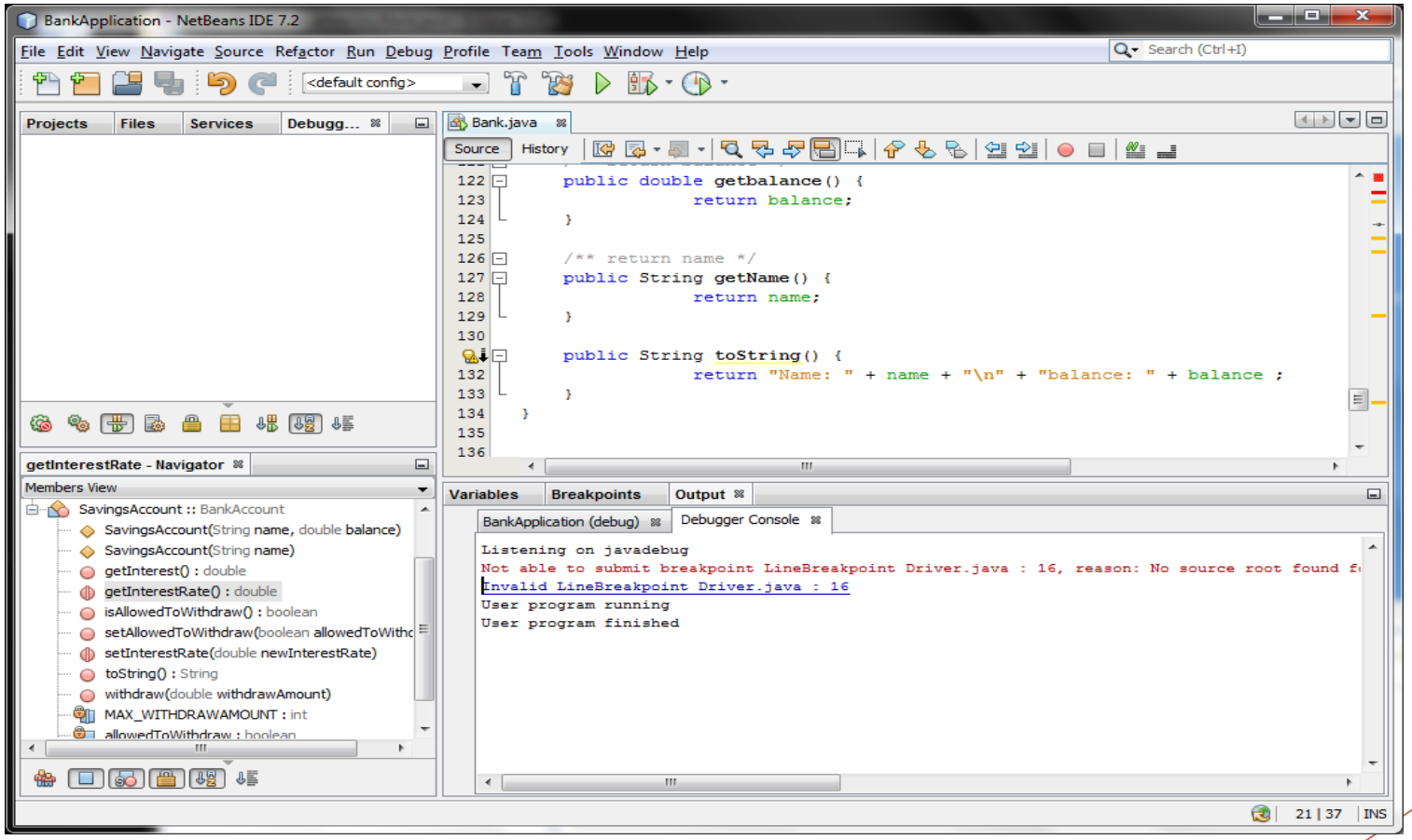
Student.java | Driver.java | Bank.java

```
        interestRate = newInterestRate;
    }

    /** withdraw money from the account with a maximum limit*/
    public void withdraw(double withdrawAmount){
        if (withdrawAmount < SavingsAccount.MAX_WITHDRAWAMOUNT && isAllowedToWithdraw()){
            super.withdraw(withdrawAmount);
        }
        else{
            System.out.println("The withdraw amount exceeds the limit of maximum withdraw amount allowed!");
        }
    }

    public String toString() {
        return super.toString() + "\nInterest Rate : " + interestRate + "\n";
    }
    public boolean isAllowedToWithdraw() {
        return allowedToWithdraw;
```

Console | Tasks

Debug using NetBeans

- Basic features are more or less the same as in Eclipse.



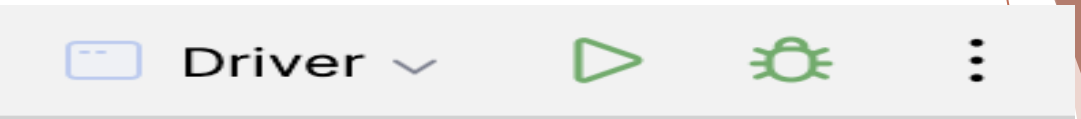
Debug using NetBeans cont'd.

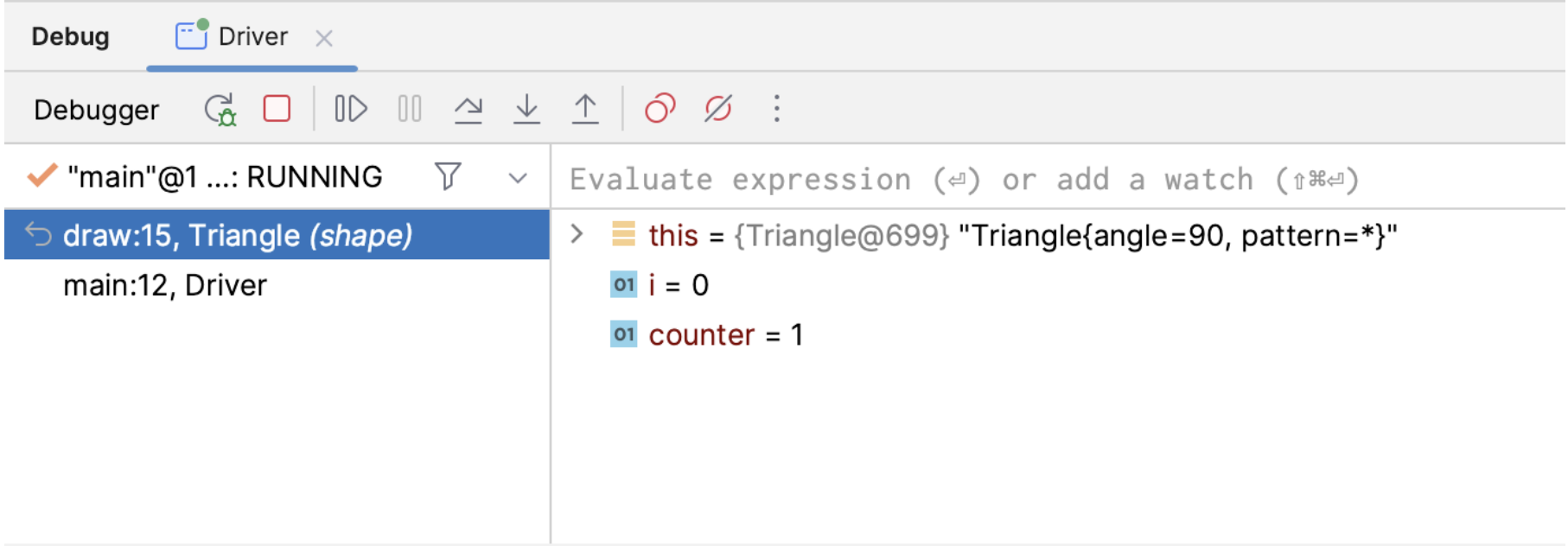
- NetBeans also supports:
  - Line breakpoints.
  - Watchpoints
  - Method breakpoints

- Provide different views:
  - Code execution
  - Variables
  - Breakpoints

IntelliJ debug panel

Run Debug





Moral of the story...

- Try to avoid writing buggy code!
  - Design your program before putting it in to code.
  - Modularize your code.
  - Test continuously what has implemented so far.

Yet ... In reality, bug free code is a hard guarantee!

References

- http://en.wikipedia.org/wiki/Debugging
- http://www.vogella.com/articles/EclipseDebugging/article.html#debugging_overview
- http://en.wikipedia.org/wiki/List_of_software_bugs