SE 2AA4, CS 2ME3 (Introduction to Software Development)

Winter 2018

# 24 Generics and Interfaces in Java DRAFT

Dr. Spencer Smith

Faculty of Engineering, McMaster University

December 15, 2017

# 24 Generics and Interfaces in Java DRAFT

- Administrative details
- Template modules and modules in Java
- Exceptions in Java
- Generic modules in Java, Set example
- Set of Integers
- Set of VectorT
- Override equals
- Interfaces
- Generic Set with Comparable
- VectorT with Comparable

# Administrative Details

TBD

# Modules in Java

- An ADT represented by a template module in the MIS becomes a class with non static members and methods (operations on the objects)
- An abstract object uses static members and methods (operations on the class)
- The name of the ADT exported type in the MIS becomes
  - The name of the class
  - The name of the file that describes the class
  - The name of the constructor(s)
- An abstract object does not have a constructor

# Constants in Java Modules

- In Java a constant should be identified as an immutable initialized class or interface field
  - `final static int ZERO = 0;`
- In Java a global constant should be specified as a public, immutable, initialized class or interface field
  - `public final static int ZERO = 0;`
- As usual, the convention is to write the name of constants in ALL_CAPS

# State Variables and Access Programs

- MIS state variables become members of the objects or of the class
- Access programs become methods of the objects or of the class
- Each object will have its own state
- No access methods can be called before constructing an object
- The constructor cannot be called twice on the same object
- The word self in the MIS becomes `this` in Java

# Exceptions in Java

- The name of the Exception is given in the MIS
- Each exception will be its own class in its own file
- Exceptions will inherit from Exception (checked) or from RuntimeException (unchecked)
- Checked exceptions
  - Exceptional cond. that can be recovered from
  - Errors caused by the user or the environment
  - Nonexistent file errors, invalid url etc.
  - Need to be caught (tell compiler what you will do), compiler fails if they are not
- Unchecked exceptions
  - Exceptional cond. that cannot (generally) be recovered from
  - Internal to the application, caused by programmers
  - Divide by zero, pushing onto a full stack, etc.
  - Can be caught, but compiler does not complain

# Example Exception I

```
/**
 * Author: S. Smith
 * Revised: March 5, 2017
 *
 * Description: Full exception
 */

public class FullException extends
    RuntimeException
{
    public FullException()
    {
    }
    public FullException(String reason)
```

# Example Exception II

```
    {
      super(reason);
    }
    public FullException(int full)
    {
      super("The container has reached its
          full size of " + full);
    }
}
```

# Generic Set Module Syntax

From H&S p. 83

**Generic Template Module**

Set(T)

**Exported Constants**

MAX_SIZE = 100

**Exported Access Programs**

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| ... | ... | ... | ... |

# Set Module Syntax

## Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| add | T | | MemberException, FullException |
| del | T | | NotMemberException |
| mem | T | bool | |
| siz | | int | |

# Using Java's HashSet?

- Does Java's HashSet directly implement the H&S Set?
- Does Java's Set interface match the H&S Set?
- How do we implement a generic class in Java?

# Generic Set Module I

```
/**
 * Author: S. Smith
 * Revised: Mar 5, 2017
 *
 * Description: A Set ADT with elements of
    type T
 */

import java.util.ArrayList;

public class GenericSet<T>
{
    protected ArrayList<T> s;
    public final static int MAX_SIZE = 100;
```

# Generic Set Module II

```
public GenericSet()
{
  s = new ArrayList<T>();
}

public Boolean mem(T t)
{
  return s.contains(t);
}

public void add(T t)
{
  //check exception for MAX_SIZE
  if (s.size() == MAX_SIZE)
  {
```

# Generic Set Module III

```
            throw new FullException("Cannot
                add to set.  Set is full.");
        }
        if (s.contains(t))
        {
            throw new MemberException("Cannot
                add to set.  Member already
                exists.");
        }
        s.add(t);
    }

    public void del(T t)
    {
        if (s.contains(t))
```

# Generic Set Module IV

```
        {
          throw new
              NotMemberException("Cannot
              delete the element.  Not in
              set.");
        }
        s.remove(t);
    }

    public int size()
    {
        return s.size();
    }
}
```

# Using Generic Set to Create a Set of Integers

- How do we create a set of integers from Generic Set?
- What is the key reserved word in Java that we need to use?

# Set of Integers I

```
/**
 * Author: S. Smith
 * Revised: March 4, 2017
 *
 * Description: Set of Integers
 */

public class IntegerSet extends
    GenericSet<Integer>
{
}
```

# Test Set of Integers I

```
/**
 * Author: S. Smith
 * Revised: March 4, 2017
 *
 * Description: Testing IntegerSet Class
 */

import org.junit.*;
import static org.junit.Assert.*;

public class TestIntegerSet
{

   private IntegerSet i;
```

# Test Set of Integers II

```
@Before
public void setUp()
{
   i = new IntegerSet();
}

@After
public void tearDown()
{
   i = null;
}

@Test
public void testAdd()
```

# Test Set of Integers III

```
    {
        i.add(6);
        assertEquals(1, i.size());
        assertTrue(i.mem(6));
    }

    @Test
    public void testMem()
    {
        i.add(6);
        assertTrue(i.mem(6));
        assertFalse(i.mem(5));
    }
}
```

# Digression: How JUnit location?

How do you tell javac and java where to find JUnit?

# Digression: Makefile I

```
#############################################################
# Author:          Joost Vandorp, S. Smith                  #
# Revised:         Thursday, Feb 24, 2017                   #
# Description:     "MAKEFILE"                               #
#############################################################

# Assumes JUnit is installed

JFLAGS = -g
JCLASS = -cp
    $(CLASSPATH):.:/opt/local/share/java/junit.ja
#JCLASS = -cp
    $(CLASSPATH):.:/usr/share/java/junit4.jar
    # on mills
```

# Digression: Makefile II

```
JC = javac
JVM = java
.SUFFIXES: .java .class
.java.class:
        $(JC) $(JFLAGS) $(JCLASS) $*.java

CLASSES = \
        GenericSet.java \
        IntegerSet.java \
        TestIntegerSet.java \
        FullException.java \
        MemberException.java \
        NotMemberException.java \
        VectorT.java \
        VectorTSet.java \
```

# Digression: Makefile III

```
            TestVectorTSet.java \
            AllTests.java \

MAIN = AllTests

default: classes

classes: $(CLASSES:.java=.class)

test: $(MAIN).class
        $(JVM) $(JCLASS)
            org.junit.runner.JUnitCore
            $(MAIN)

clean:
```

# Digression: Makefile IV

```
$(RM) *.class
```

# VectorT I

```
/**
 * Author: S. Smith
 * Revised: March 5, 2017
 *
 * Description: Vector ADT class
 */

import static java.lang.Math.*;

public class VectorT
{

   protected double xc;
```

# VectorT II

```
protected double yc;

public VectorT(double x, double y)
{
  xc = x;
  yc = y;
}

public double xcrd()
{
  return xc;
}

public double ycrd()
{
```

# VectorT III

```
        return yc;
    }

    public double mag()
    {
        return sqrt(pow(xc, 2.0) + pow(yc,
            2.0));
    }
}
```

# Set of VectorT I

```
/**
 * Author: S. Smith
 * Revised: March 5, 2017
 *
 * Description: Set of VectorT
 */

public class VectorTSet extends
    GenericSet<VectorT>
{
}
```

# Test Set of VectorT I

```
/**
 * Author: S. Smith
 * Revised: March 4, 2017
 *
 * Description: Testing VectorTSet Class
 */

import org.junit.*;
import static org.junit.Assert.*;

public class TestVectorTSet
{

    private VectorTSet p;
```

# Test Set of VectorT II

```
@Before
public void setUp()
{
  p = new VectorTSet();
}

@After
public void tearDown()
{
  p = null;
}

@Test
public void testAdd()
```

# Test Set of VectorT III

```
        {
            p.add(new VectorT(4, 5));
            assertEquals(1, p.size());
            //assertTrue(p.mem(new VectorT(4,
                5)));
        }
    }
```

# Why Did the Test Fail?

Why did the membership test fail for VectorT, while it worked for Integer?

# Set of VectorT with Equals I

```
/**
 * Author: S. Smith
 * Revised: March 5, 2017
 *
 * Description: Vector ADT class
 */

import java.util.Objects;
import static java.lang.Math.*;

public class VectorT
{
```

# Set of VectorT with Equals II

```
public static final double TOLERANCE =
    1e-15;

protected double xc;

protected double yc;

public VectorT(double x, double y)
{
   xc = x;
   yc = y;
}

public double xcrd()
{
```

# Set of VectorT with Equals III

```
    return xc;
  }

  public double ycrd()
  {
    return yc;
  }

  public double mag()
  {
    return sqrt(pow(xc, 2.0) + pow(yc,
        2.0));
  }

  @Override
```

# Set of VectorT with Equals IV

```
public boolean equals(Object o)
{
  if (o == this) return true;
  if (!(o instanceof VectorT))
  {
    return false;
  }
  VectorT v = (VectorT) o;
  return (abs(xc-v.xc) <= TOLERANCE) &&
     (abs(yc-v.yc) <= TOLERANCE);
}

@Override
public int hashCode()
{
```

# Set of VectorT with Equals V

```
            return Objects.hash(xc, yc);
        }
    }
```

# Interfaces

- An interface in Java provides a set of methods and their signatures, but no implementation
- Facilitates reusable solutions
  - Algorithms can be written in terms of the interface
  - The algorithm can be used for any class that implements the interface
- All methods in an interface are automatically public
- An interface does not have instance variables
- You can never construct an object of an interface type, because there is no type associated with an interface the type comes from the class that realizes the interface
- To Generic MIS syntax add type constraints, as in Ghezzi et al.

# Generic Set Module Syntax

Using Ghezzi notation for generic modules

**Generic Module**

Set(T with equals: $T \times T \rightarrow$ Boolean, compareTo: $T \times T \rightarrow$ int)

**Exported Constants**

MAX_SIZE = 100

**Exported Access Programs**

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| ... | ... | ... | ... |

# Generic Set with Comparable I

```
/**
 * Author: S. Smith
 * Revised: Mar 5, 2017
 *
 * Description: A Set ADT with elements of
    type T
 */

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

public class GenericSet<T extends
    Comparable<T>>
```

# Generic Set with Comparable II

```
{
    protected ArrayList<T> s;
    public final static int MAX_SIZE = 100;
    public GenericSet()
    {
      s = new ArrayList<T>();
    }

    public Boolean mem(T t)
    {
      return s.contains(t);
    }

    public void add(T t)
    {
```

# Generic Set with Comparable III

```
        //check exception for MAX_SIZE
        if (s.size() == MAX_SIZE)
        {
            throw new FullException("Cannot
                add to set.  Set is full.");
        }
        if (s.contains(t))
        {
            throw new MemberException("Cannot
                add to set.  Member already
                exists.");
        }
        s.add(t);
    }
```

# Generic Set with Comparable IV

```
public void del(T t)
{
  if (s.contains(t))
  {
    throw new
      NotMemberException("Cannot
      delete the element.  Not in
      set.");
  }
  s.remove(t);
}

public int size()
{
  return s.size();
```

# Generic Set with Comparable V

```
    }

//    public T max()
//    {
//       return Collections.max(s);
//    }

    public T max()
    { //should have exception for empty
      T tmp = s.get(0);
      for(T x: s)
      {
        if (tmp.compareTo(x) == -1)
        {
          tmp = x;
```

# Generic Set with Comparable VI

```
            }
        }
        return tmp;
    }

}
```

# What Next?

- What do we have to do with Integer to get it to work with our Set with max?
- What do we have to do with VectorT to get it to work with our Set with max?

# VectorT implements Comparable I

```
/**
 * Author: S. Smith
 * Revised: March 5, 2017
 *
 * Description: Vector ADT class
 */

import java.util.Objects;
import static java.lang.Math.*;

public class VectorT implements
    Comparable<VectorT>
{
```

# VectorT implements Comparable II

```
public static final double TOLERANCE =
    1e-15;

protected double xc;

protected double yc;

public VectorT(double x, double y)
{
   xc = x;
   yc = y;
}

public double xcrd()
{
```

# VectorT implements Comparable III

```
    return xc;
  }

  public double ycrd()
  {
    return yc;
  }

  public double mag()
  {
    return sqrt(pow(xc, 2.0) + pow(yc,
        2.0));
  }

  @Override
```

# VectorT implements Comparable IV

```
public boolean equals(Object o)
{
  if (o == this) return true;
  if (!(o instanceof VectorT))
  {
    return false;
  }
  VectorT v = (VectorT) o;
  return (abs(xc-v.xc) <= TOLERANCE) &&
     (abs(yc-v.yc) <= TOLERANCE);
}

@Override
public int hashCode()
{
```

# VectorT implements Comparable V

```
    return Objects.hash(xc, yc);
  }

  @Override
  public int compareTo(VectorT v)
  {
    double left = this.mag();
    double right = v.mag();
    if (left < right)
    {
      return -1;
    }
    if (left > right)
    {
      return 1;
```

# VectorT implements Comparable VI

```
            }
            return 0;
        }
    }
```