Java Generics Examples and Exercises

CSCI 2300

Review

- POGIL activity on Generics and Collections
- Key lessons from that activity
    - ?

Creating your own Generic types

- Only if it makes sense
- Repeated behaviors on different object types
- Data structures
    - Linked list
    - Binary tree
    - Priority queue
    - Etc

Example

- Consider a Matrix class

$$\begin{matrix} 3 & 2 & 1 \\ 2 & 5 & 8 \\ 7 & 8 & 8 \end{matrix}$$

- Implementation options
    - Use 2-dimentional array of `double`
    - Use 2-dimentional array of `Object`
    - Use generic type

Implementing Matrix with 2-dimentional array of `double`

```
public class Matrix
{
   private double [][]values;
   private int rows;
   private int cols;
   public Matrix(int r, int c)
   {
      rows = r;
      cols = c;
      values = new double[rows][cols];
   }
   public void set(int r, int c, double v){values[r][c] = v;}
   public double get(int r, int c){return values[r][c];}
}
```

Using Matrix class

```
Matrix m = new Matrix(3, 3);
m.set(0, 0, 1.0);
double v = m.get(0,0);
```

Suppose we have the following Matrix object.
Can we use it to store a matrix of boolean values?

Matrix m = new Matrix(3,3);

A. We can set a boolean value in `Matrix m` with `m.set(0, 0, true)`, and true will get stored as double.
B. We can use double to represent booleans, where 0 is false and non-zero is true
C. We cannot use Matrix class to represent a matrix of booleans
D. A & B

What if we wanted a Matrix of non-numeric values?

```
public class Matrix
{
   private Object [][]values;
   private int rows;
   private int cols;
   public Matrix(int r, int c)
   {
      rows = r;
      cols = c;
      values = new Object[rows][cols];
   }
   public void set(int r, int c, Object v){values[r][c] = v;}
   public Object get(int r, int c){return values[r][c];}
}
```

Suppose we have a class Card and the the following Matrix object. Can we use it to store a matrix of Card objects?

Matrix m = new Matrix(3,3);

A. We can set a `Card c` object in `Matrix m` with `m.set(0, 0, c)` because Card extends Object
B. We can up-cast `Card c` to `Object` and store it in matrix with `m.set(0, 0, (Object)c)`
C. We cannot use Matrix class to represent a matrix of Card objects
D. A & B

Suppose we added a 'get' method to Matrix

```
public Object get(row r, col c)
{
    return values[r][c];
}
```

Can we use the get method to retrieve a card from a matrix?

A. Yes, but we need to down-cast it to Card type: Card c = (Card)m.get(0, 0)
B. Yes, you can simply call the get method: Card c = m.get(0,0);
C. No, you cannot assign Object reference returned from Matrix to a Card reference.
D. A & B

Using Generics

```
public class Matrix<T>
{
   private T [][]values;
   private int rows;
   private int cols;
   public Matrix(int r, int c)
   {
      rows = r;
      cols = c;
      values = new T[rows][cols]; // ← THIS WILL NOT COMPILE
   }
   public void set(int r, int c, T v){values[r][c] = v;}
   public T get(int r, int c){return values[r][c];}
}
```

Java Type Erasure

- Types are checked at compile time
- Replaces all type parameters in generic types with Object (or type bounds)
  - Produced byte code contains only ordinary classes, interfaces, methods
- Inserts type casts, when necessary
- Only one class is generated
  - Different from C++

Generic type bounds

```
public class Matrix<T extends Object>
{
   private Object [][]values;
   private int rows;
   private int cols;
   public Matrix(int r, int c)
   {
      rows = r;
      cols = c;
      values = new Object[rows][cols]; // ← COMPILER WARNING
   }
   public void set(int r, int c, T v){values[r][c] = v;}
   public T get(int r, int c){return (T)values[r][c];}
}
```

Other Alternatives

- Use Java Collections
  - List
  - Array List

Lab 12

- Create a generic type to represent a pair of objects. The two objects in the pair can be of different types
- Example usage: Pair<Integer, Card> p = new Pair<Integer, Card>(…);