Expressions Collections, Generics

Tutorial 5 (24th February 2021)



Lists

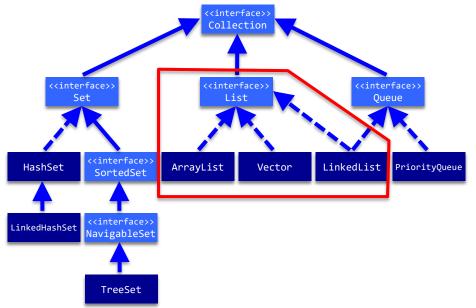
interface **List** is an extension of **Collection**

- List adds methods to manipulate elements via indices
 - void add(int index, E element)
 - E get(int index)
 - E remove (int index)
 - E set (int index, E element) ...

ArrayList implements the interface **List**

other implementations are

LinkedList and Vector



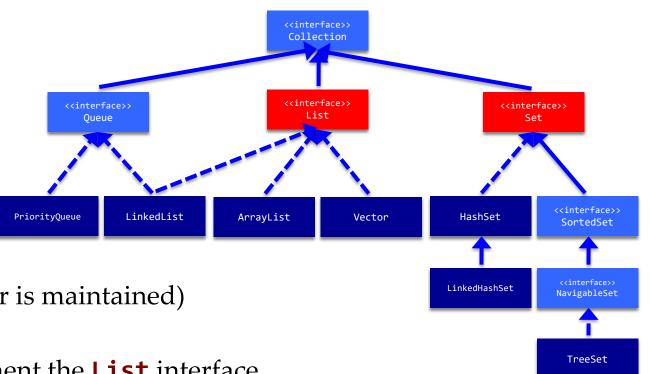
collection relationships

Set

- does not contain duplicates
- can (sometimes) be sorted!

List

- elements are ordered (insertion order is maintained)
- elements can occur more than once
- ArrayList and LinkedList implement the List interface
 - next lecture we will discuss the differences between ArrayList and LinkedList
- Vector is very similar to ArrayList in API,
 vectors are thread-safe and hence somewhat slower



the class Collections

do not confuse it with the interface Collection



- contains algorithms for collections
- like **Arrays** for arrays

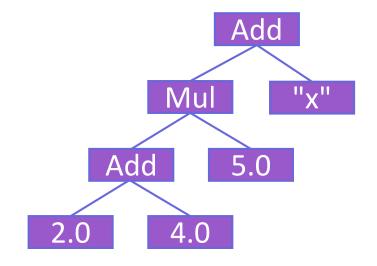
implemented algorithms:

```
sort, binarySearch, reverse, shuffle,
fill, copy, min, max, addAll,
frequency, disjoint
```

three ways to access all list elements

```
for (int i = 0; i < list.size(); i += 1) {</pre>
                                                        + any order possible
    Card card = list.get(i);
                                                        – get(i) can be inefficient
    if (card.face == Card.Face.Queen) {
         System.out.println("Queen1: " + card);
                                                         + compact
                                                          + efficient
          for (Card card : list) {
                                                          list cannot be changed
              if (card.face == Card.Face.Queen) {
                   System.out.println("Queen2: " + card);
                            Iterator<Card> iter = list.iterator();
                            while (iter.hasNext()) {
                                Card card = iter.next();
+ efficient
                                if (card.face == Card.Face.Queen) {
+ flexible
+ not restricted to loop
                                     System.out.println("Queen3: " + card);
! remove only last item
                                                             Radboud Univers
```

Datatype for expressions

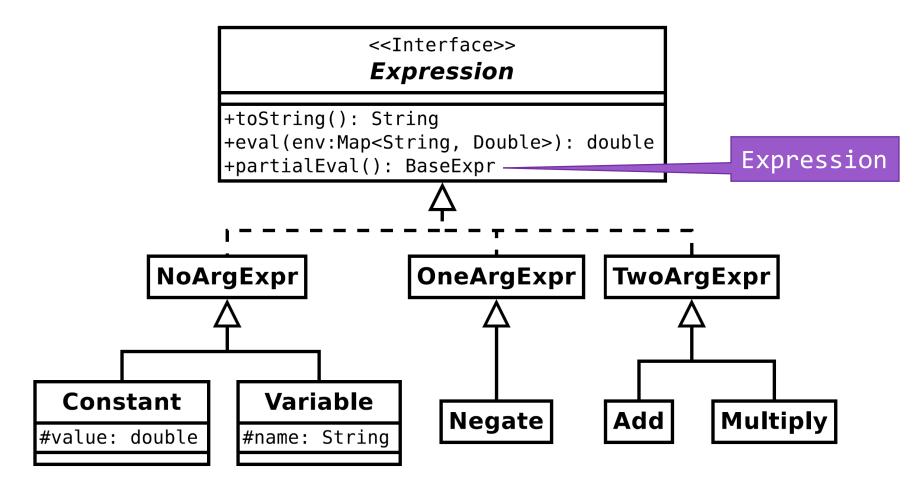


operations:

- toString: ((2 + 4) * 5 + x)
- evaluate: with $x \mapsto 12$ this yields 42
- optimise (evaluate parts that do not contain variables)



Types for expressions



evaluation

- To evaluate an expression we need an environment assigning values to variables.
- The eval method has the environment as a parameter.
- An environment can be implemented in Java using a Map:
 - Map is a generic interface:

Interface Map<K,V>

Type Parameters:

K - the type of keys maintained by this map

V - the type of mapped values



interface Map<K,V>

Melander Marie Mar	Method Summary	
### Common of the manager from this and popular department. ### Common of the manager from this and popular department. ### Common of the manager for the special for the manager for the buyer. ###################################	All Methods Instance Methods Abstract Methods Default Methods	
interest of the mappings from the same perposed spectroal spectral SP compagning from the S	Modifier and Type	Method and Description
Heavil V Comment to compute a suspension from the specified leve and norm entanges when they not lift affect in so contracting complete in the specified leve and norm contracting when the profession and contracting comments are the specified leve and norm of the specified leve and norm of the specified leve and norm analyses are manageng great theory and sea comments are the specified leve. Broken Comments of the specified leve, and suspense from the specified leve. Broken Comments of the specified leve, and suspense from the specified leve. Broken Comments of the specified leve. Broken Comments	void	
In the specified by in an indexed and such as when for an image that mility and passed and a	default V	
If the value for the specified ker is premeted ados multi, stempts to compute a sere mapping given the keys and its current mapped value. Bottoms from fifth map contains an engipting for the specified keys. Bottoms from fifth map pass once or more keys to the specified keys. Between from fifth map pass once or more keys to the specified keys. Bottoms after twen of the anappage contained in this map. Bottoms after twen of the anappage contained in this map. Bottoms after twen of the anappage contained in this map. Bottoms after twen of the anappage contained in this map. For anappage contained in this map. Bottoms after twen the map for equality of the keys of the decide fifthers an exception. For anappage contained in this map, or man an anapping for the keys of the decide fifthers an exception. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man an anapping for the keys. For anappage contained in this map, or man and page or man and page or man and page or man anapping for the keys. For anappage contained in this map, or man anapping for the key	default V	
Remark Teer d'ala mage commans en angegone despected skey. Referre for tree l'ala mage commans en angegone de l'anceptible value. Referre for tree l'ala mage morate our more leys to the specified value. Referre for tree l'ala mage morate our more leys to the specified value. Referre for tree l'ala mage morate our protection de personal que personal que personal que l'acceptant que l'ala mage commans de l'ala mage commans de l'ala mage. Referre for tree de legre value du mage. Van de l'ala mage de l'ala mage de l'ala mage de l'ala mage. Van de l'ala mage de l'ala mage de l'ala mage. Referre for tree de les protected objet va thit in mage de maint is non an parque for the sion throws an mergrison. Van de l'ala mage de l'ala mage de l'ala mage commans son mages prote de soits theore an mergrison. Van de l'ala mage de l'ala mage commans son mages prote de soits throws an mergrison. Referre for tree de l'ala mage commans son mages prote de soits throws an mergrison. Referre for tree de l'ala mage commans son mages prote de soits throws an mergrison. Referre for tree de l'ala mage commans son mages prote de soits throws an mergrison. Referre for tree de l'ala mage commans son mages prote de soits throws an mergrison. Referre for tree de l'ala mage commans son mages prote les less. Referre for tree de l'ala mage commans son mages prote les less. Referre for tree de l'ala mage commans son mages protection. Referre for tree de l'ala mage commans son mages protection. Referre for tree de l'ala mage commans son mages protection. Referre for tree de l'ala mage commans son mages protection. Referre for tree de l'ala mage commans son mages protection. Referre for tree de l'ala mage commans son mages protection. Referre for tree personal de presentation. Referre for tree personal de perso	default V	
Interest Post (In many many one or more levys to the specified value. Set-files, Estray, Kryo Annual (Sept. 1) Annual (Sept. 2) Annual	boolean	
Setums 45 tive of the mappings contained in this map. beloated sequal (abject of compares the specified object with this map for equality. default vaid of radial (vaid object with this map for equality. For first the specified object with this map for equality. Priffers the great active in the map until all entries have been processed or the action throws an acception. Priffers the great active in the map until all entries have been processed or the action throws an acception. Priffers the great active in the map until all entries have been processed or the action throws an acception. Priffers the great active in the map until all entries have been processed or the action throws an acception. Priffers the great active in the map until all entries have been processed or the action throws an acception. Priffers the great active in the map until all entries have been processed or the action throws an acceptance. Priffers the great active in the great active in the great active in this map contains no mapping for the key. Proffers the great active is mapped or default Value if this map contains no hely value enspired. Priffers the great active with the great active in this map contains no hely value enspired. Priffers the great active with the great active in this map contains no hely value enspired. Priffers the great was of the keys contained in this map. Priffers the specified whise with the specified whise with the specified whise or in associated with null, associates is with the green normal value. Priffers the specified whise with the specified key in this map (optional operation). Priffers the specified whise with the specified key in this map (optional operation). Priffers the specified whise with a supplied for its map (optional operation). Priffers the specified whise with a supplied for its interest in with the green nation until element in the current value. Priffers the specified whise with the specified walue. Priffers the specified walue with a value or is associated with a va	boolean	
Compare the specified object with this map fore equality. for facila life foreigness? spier f. S. spies A set 100 performant contains to map mind all entries have been processed or the action throws an exception. Performant for which the specified key is mapped, or null if this map contains no mapping for the key. Beautiful four which the specified key is mapped, or null if this map contains no mapping for the key. Performant contains to which the specified key is mapped, or null if this map contains no mapping for the key. Performant contains to which the specified key is mapped, or null if this map contains no mapping for the key. Performant contains to which the specified key is mapped, or null if this map contains no mapping for the key. Performant contains to which the specified key is mapped, or null if this map contains no mapping for the key. Performant contains to which the specified key is mapped, or null if this map contains no mapping for the key. Performant contains to which the specified key is mapped, or null if this map contains no mapping for the key. Performant contains no which the specified key is mapped. Performant contains no key value mappings. Performant contains no key value for this map contains no mappings for the key. Performant contains no key value mappings. Performant contain	Set <map.entry<k,v>></map.entry<k,v>	
Performs the given action for each eatiny in this map until al entries have been processed or the action throw an exception. Performs the value to which the specified key is mapped, or full if this map contains no mapping for the key. Performs the value to which the specified key is mapped, or default Value if this map contains no mapping for the key. Performs the value to which the specified key is mapped, or default Value if this map contains no mapping for the key. Performs the value to which the specified key is mapped, or default Value if this map contains no key value mappings. Performs the value or value for this map. Performs the value or value for this map. Performs the value or value propriets. P	boolean	
the fault V in the pecified key is mapped, or not it fits map contains no mapping for the key. the fault V in the value to which the specified key is mapped, or defaultValue if this map contains no mapping for the key. the value to which the specified key is mapped, or defaultValue if this map contains no mapping for the key. the value to which the specified key is mapped, or defaultValue if this map contains no mapping for the key. the fault V is the specified key is mapped, or defaultValue if this map contains no mapping for the key. the fault V is the specified key is mapped, or defaultValue if this map contains no mapping for the key. the fault V is the specified key is mapped, or defaultValue if this map contains no mapping for the key. the fault V is the specified key is mapped, or defaultValue if this map contains no mapping for the key. the fault V is the specified key is mapped and this map. the specified key is mapping. the specified key is value, Bifunctions? super V,7 super V,7 extends V> recappingFunction) and the specified key is walue and value or is associated with a value for is associated with an ill, associates it with the given non-null value. the specified value with the specified key in this map (optional operation). the fault V is the specified key in the specified key in this map (optional operation). the specified key is not already associated with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value. the specified key is not already associated with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value. the specified key is not already associated with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value. The specified key is not already associated with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value. The specified key is	default void	
Factors the value to which he specified key is mapped, or de fault Value if this map contains no mapping for the key. Associated Associated Associated in this map. Associated in this map. Associated with a value on paining not key value mappings. Associated with a value of paining not key value mappings. Associated with a value of a sasociated with null, associates it with the given non-null value. Associates the specified key in this map (optional operation). Associates the specified key in this map (optional operation). Associates the specified value with the specified key in this map (optional operation). Associates the specified value with the specified key in this map (optional operation). Associates the specified value with the specified key in this map (optional operation). Associates the specified key in this map (optional operation). Copies all of the mappings from the specified map to this map (optional operation). Associates the specified key in this map (optional operation). Feave (Object key) Freshy (Object key) Associates the support of a key from this map if it is present (optional operation). Associates the mapping for a key from this map if it is present (optional operation). Associates the mapping of a key from this map if it is present (optional operation). Associates the unity of the Specified key only if it is currently mapped to the specified value. Associates the unity of the Specified key only if it is currently mapped to the specified value. Associates the unity of the Specified key only if it is currently mapped to some value. Associates the unity of the Specified key only if it is currently mapped to the specified value. Associates the unity of the Specified key only if it is currently mapped to the specified value. Associates the unity of the Specified key only if it is currently mapped to the specified value. Associates the unity of the Specified key only if it is currently mapped to the specified value. Associates the unity of the Specified key only if	V	
botean command to the hash code value for this map. botean command to the keys contained in this map. befault V and command to the keys contained in this map. befault V and command to the keys contained in this map. befault V and command to the keys contained in this map. befault V and command to the keys contained when value or is associated with a value or is associated with the given non mull value. befault V and command to the specified key is not already associated with a value or is associated with the given non mull value. befault V and command to the specified value with the specified key in this map (optional operation). befault V and command to the mappings from the specified was not already associated with a value or is mapped to null) associates it with the given value and returns null, else returns the current value. befault V and command to the mapping from the specified key in this map (optional operation). befault V and command to the mapping from the specified will we will associates it with the given value and returns null, else returns the current value. befault V and command to the mapping from the specified key only if it is currently mapped to null) associates it with the given value and returns null, else returns the current value. befault V and command to the mapping from the specified key only if it is currently mapped to null, associates it with the given value and returns null, else returns the current value. befault V and command to the mapping from the specified key only if it is currently mapped to null, associates it with the given value and returns null, else returns the current value. befault volt befault befau	default V	
Returns 1 ret if this map contains no key-value mappings. Set-Ko Reget (int	
Herma Set view of the keys contained in this map. default V presq (K key, V value) sascoitated with a value or is associated with null, associates it with the given non-null value. prof (K key, V value) sascoitate the specified key in this map (optional operation). default V prof (K key, V value) sascoitate with mis map (optional operation). default V prof (K key, V value) sascoitate with expecified map to this map (optional operation). prof (K key, V value) sascoitate with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value. prov (Object key) sascoitate with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value. prov (Object key) sascoitate with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value. prov (Object key) sascoitate with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value. prov (Object key) sascoitate with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value. prov (Object key) sascoitate with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value. prov (Object key) sascoitate value (or is mapped to null) associates it with the given value and returns null, else returns the current value. prov (Object key) sascoitate value (or is mapped to null) associates it with the given value and returns null, else returns the current value. prov (Object key) sascoitate value (or is mapped to null) associates it with the given value and returns null, else returns the current value. prov (Object key) sascoitate value (or is mapped to null) associates it with the given value and returns null, else retur	boolean	
If the specified key is not already associated with null, associates it with the given non-null value. Put (K key, V value)	Set <k></k>	
Associates the specified value with the specified key in this map (optional operation). void 200 putfl/Kipps/ extends K,? extends V, or value or in mapped to nutly associates it with the given value and returns nutl, else returns the current value. V remove(Object Key) Removes the mapping for a key from this map if it is present (optional operation). default V molean default V molean default V molean default void molean default boolean default bo	default V	
default V remove (Object key, Object value) Removes the mapping from the specified key only if its currently mapped to the specified value. remove (Object key, Object value) Removes the entry for the specified key only if its currently mapped to the specified value. default V remove (Object key, Object value) Removes the entry for the specified key only if its currently mapped to the specified value. default boolean replace (K key, V value) Replaces the entry for the specified key only if it is currently mapped to some value. default void replace (K key, V voldValue, V newValue) Replaces the entry for the specified key only if currently mapped to the specified value. replace (K key, V voldValue, V newValue) Replaces the entry for the specified key only if currently mapped to the specified value. replace (K key, V voldValue, V newValue) Replaces each entry's value with the result only happed to the specified value. replace seach entry's value with the result only for the specified value. replace seach entry's value with the result only key function) Replaces each entry's value with the result only key function) Replaces each entry's value with the result only key function) Replaces each entry's value with the result only key function) Replaces each entry's value with the result only key function) Replaces each entry's value with the result only key function) Replaces each entry's value with the result on the entry until all entries have been processed or the function throws an exception.	v	
If the specified key is not already associated with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value. Y remove (Object key) Removes the mapping for a key from this map if it is present (optional operation). default boolean removes the entry for the specified key only if it is currently mapped to the specified value. default V Removes the entry for the specified key only if it is currently mapped to some value. default boolean replace(K key, V oldValue, V newValue) Replaces the entry for the specified key only if unerturny mapped to the specified value. default void replace(K key, V oldValue, V newValue) Replaces the entry for the specified key only if currently mapped to the specified value. replace(K key, V oldValue, V newValue) Replaces each entry's value with the result of invoking the given function) Replaces each entry's value with the result of invoking the given function on that entry until all entries have been processed or the function throws an exception.	void	
Removes the mapping for a key from this map if it is present (optional operation). default boolean remove (Deject key, Object value) Removes the entry for the specified key only if it is currently mapped to the specified value. default V default boolean replace(K key, V value) Replaces the entry for the specified key only if it is currently mapped to some value. default void replace (K key, V oldValue, V newValue) Replaces the entry for the specified key only if currently mapped to the specified value. default void replace (K key, V oldValue, V newValue) Replaces each entry's value with the result of invoking the given function on that entry until all entries have been processed or the function throws an exception. int	default V	
Removes the entry for the specified key only if it is currently mapped to the specified value. default V replace (K key, V value) Replaces the entry for the specified key only if it is currently mapped to some value. default boolean default void default void replace (K key, V oldWalue, V newWalue) Replaces the entry for the specified key only if currently mapped to the specified value. default void replace (K key, V oldWalue, V newWalue) Replaces the entry for the specified key only if currently mapped to the specified value. default void seplaces the entry for the specified key only if it is currently mapped to some value. default void seplaces the entry for the specified key only if it is currently mapped to some value. seplaces the entry for the specified key only if it is currently mapped to some value. default void seplaces the entry for the specified key only if it is currently mapped to some value. default void seplaces the entry for the specified key only if it is currently mapped to some value. default void seplaces the entry for the specified key only if it is currently mapped to some value. default void seplaces the entry for the specified key only if it is currently mapped to some value. default boolean seplaces the entry for the specified key only if it is currently mapped to some value. default boolean seplaces the entry for the specified key only if it is currently mapped to some value. default boolean seplaces the entry for the specified key only if it is currently mapped to some value. default boolean seplaces the entry for the specified key only if it is currently mapped to some value.	v	
Replaces the entry for the specified key only if it is currently mapped to some value. default boolean replace(K key, V oldValue, V newValue) Replaces the entry for the specified key only if currently mapped to the specified value. default void replaces each entry's value with the result of invoking the given function on that entry until all entries have been processed or the function throws an exception. int size()	default boolean	
Replaces the entry for the specified key only if currently mapped to the specified value. default void replaceAl(BiFunction super K,? super V,? extends V function) Replaces each entry's value with the result of invoking the given function on that entry until all entries have been processed or the function throws an exception. size()	default V	
Replaces each entry's value with the result of invoking the given function on that entry until all entries have been processed or the function throws an exception. size()	default boolean	
	default void	
	int	
Collection<∜> values() Returns a Collection view of the values contained in this map.	Collection <v></v>	



Java map

```
public interface Map<K,V> {
  V get( Object key );
  V put( K key, V value );
a commonly used implementation
public class HashMap<K,V> implements Map<K,V> {
Example
Map<String, Double> env = new HashMap<>();
env.put("x", 7.0);
env.put("a", 42.0);
double a = env.get("a");
                                   Double is a class (reference type)
                                   double is a primitive type
```

double vs. Double

- double d = 3.14;
- sometimes we need an object instead of a primitive value
 e.g. in the Map from names to values
- Java provides standard boxed variants of primitive (unboxed) types
- Double db = new Double(2.78);
- class Double contains also convenience methods (parsing, ..)
- instances of Double are immutable (like strings).
- Java performs *auto boxing* and *auto unboxing*: a silent conversion between double and Double

```
Double db = 2.78; // auto boxing
db = db * 2.0; // which is equivalent to:
db = new Double( db.doubleValue() * 2.0 );
```



partial evaluation (optimization)

- evaluate the parts of an expression that do not depend on the values of variables
- rules:

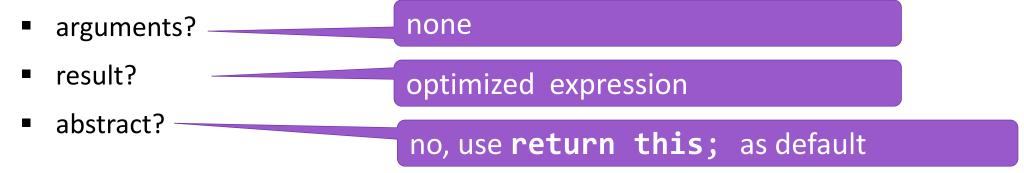
$$n+m$$
 \rightarrow o where $o=n+m$
 $e+0$ \rightarrow e
 $0+e$ \rightarrow e
 $n\times m$ \rightarrow o where $o=n\times m$
 $0\times e$ \rightarrow 0
 $1\times e$ \rightarrow e
 $e\times 0$ \rightarrow 0
 $e\times 1$ \rightarrow e
 $neg(n)$ \rightarrow $-n$

n, *m*: numerical constantse: an arbitrary expression



implementing partial evaluation

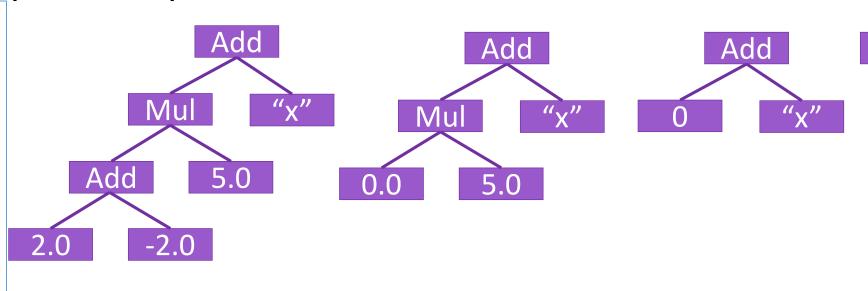
Expression has a method partialEval for partial evaluation



- override this method to apply the rules (if this is possible)
 - how do we check if an argument is some constant?
 - polymorphism?
 - special method that yields value if it is there! what is the result if there is no value?

bottom-up or top-down?

```
add(n, m) \rightarrow n+m
add(e, 0) \rightarrow e
add(0, e) \rightarrow e
mul(n, m) \rightarrow n\times m
mul(0, e) \rightarrow 0
mul(1, e) \rightarrow e
mul(e, 0) \rightarrow 0
mul(e, 1) \rightarrow e
neg(n) \rightarrow -n
```



evalutate arguments first (bottom-up)!



Representing binary expressions

```
public abstract class TwoArgExpr implements Expression {
  private final Expression x, y;
  public TwoArgExpr( Expression x, Expression y ) {
    this.x = x;
                                                                                    <<Interface>>
    this.y = y;
                                                                                    Expression
                                                                            +toString(): String
                                                                            +eval(env:Map<String, Double>): double
                                                                            +partialEval(): BaseExpr
  public Expression getX() {
       return x;
                                                                        NoArgExpr
                                                                                       OneArgExpr
                                                                                                   TwoArgExpr
  public Expression getY() {
                                                                              Variable
                                                                  Constant
                                                                                         Negate
                                                                                                   Add
                                                                                                         Multiply
       return y;
                                                                             #name: String
                                                                 #value: double
```

Binary expressions: Add

```
<<Interface>>
public class Add extends TwoArgExpr {
                                                                           Expression
  public Add( Expression x, Expression y ) {
                                                                   +toString(): String
    super( x, y );
                                                                   +eval(env:Map<String, Double>): double
                                                                   +partialEval(): BaseExpr
  @Override
                                                               NoArgExpr
                                                                              OneArgExpr
                                                                                         TwoArgExpr
  public String toString() {
    return "(" + getX() + "+" + getY() + ")";
                                                         Constant
                                                                     Variable
                                                                                         Add
                                                                                Negate
                                                        #value: double
                                                                    #name: String
  @Override
  public double eval( Map<String, Double> env ) {
    return getX().eval(env) + getY().eval(env);
  @Override
  public Expression partialEval() { ... }
```

Multiply

Binary expressions: Mul

```
public class Mul extends TwoArgExpr {
                                                                             <<Interface>>
  public Mul ( Expression x, Expression y ) {
                                                                             Expression
    super (x, y);
                                                                     +toString(): String
                                                                     +eval(env:Map<String, Double>): double
                                                                     +partialEval(): BaseExpr
  @Override
                                                                                           TwoArgExpr
                                                                 NoArgExpr
                                                                                OneArgExpr
  public String toString() {
    return "(" + getX() + "*" + getY() + ")";
                                                           Constant
                                                                       Variable
                                                                                  Negate
                                                                                           Add
                                                                                                 Multiply
                                                          #value: double
                                                                      #name: String
  @Override
  public double eval( Map<String, Double> env ) {
    return getX().eval( env ) * getY().eval( env );
  @Override
  public Expression partialEval() { ... }
```

Code duplication:

```
public class Add extends TwoArgExpr {
  public Add( Expression x, Expression y ) {
    super ( x, y );
  @Override
  public String toString() {
   return "(" + getX() + "+" + getY() + ")";
  @Override
  public double eval( Map<String, Double> env ) {
   return getX().eval(env) + getY().eval(env);
  @Override
  public Expression partialEval() { ... }
```

```
ctends TwoArgExpr {
```

```
public class Mul extends TwoArgExpr {
 public Mul ( Expression x, Expression y ) {
    super ( x, y );
 @Override
 public String toString() {
   return "(" + getX() + "*" + getY() + ")";
 @Override
 public double eval( Map<String, Double> env ) {
    return getX().eval(env) * getY().eval(env);
 @Override
 public Expression partialEval() { ... }
```

Solution: Pull the method up

Move duplicate code to the super class

public abstract class TwoArgExpr implements Expression {

Use the strategy pattern to implement different behaviour

```
predefined generic
      interface
BinaryOperator<T>
```

```
T apply(T x,T y)
```

```
private final Expression x, y;
private final BinaryOperator<Double> evalOp;
private final BinaryOperator<Expression> peOp;
public TwoArgExpr( Expression x,Expression y,BinaryOperator<Double> evalOp,BinaryOperator<Expression> peOp ) {...}
@Override
public double eval( Map<String, Double> env ) {
    return evalOp.apply( x.eval(env), y.eval(env) );
@Override
public Expression partialEval() {
    return peOp.apply( x.partialEval(), y.partialEval() );
@Override
public String toString() {
   return "(" + x + getOperator() + y + ")";
protected abstract String getOperator();
```



Binary expressions: Add revisited

```
public class Add extends TwoArgExpr {
  public Add( Expression x, Expression y ) {
    super( x, y );
 @Override
  public String toString() {
    return "(" + getX() + "+" + getY() + ")";
 @Override
 public double eval( Map<String, Double> env ) {
    return getX().eval(env) + getY().eval(env);
 @Override
 public Expression partialEval() { ... }
```



```
public class Add extends TwoArgExpr {
  public Add( Expression x, Expression y ) {
    super( x, y, ???, ??? );
  }

  private Expression partialEvalAdd(...) { ... }
}
```

Functional interfaces

- A functional interface is an interface with a single abstract method
 - It can be implemented by a lambda expression (more about lambda expressions next week)
- Instead of

```
public class AddEvaluator implements BinaryOperator<Double> {
   @Override
   public Double apply( Double t, Double u ) {
       return t + u;
public Add( Expression x, Expression y ) {
    super ( x, y, new AddEvaluator (), ... ),
 we can write (omitting the definition of class AddEvaluator)
   public Add( Expression x, Expression y ) {
```

Constructor of the class Add

Constructor of TwoArgExpr

lambda expression

```
super ( x, y, ( a1, a2 ) -> a1 + a2, ... ),
```

Binary expressions: Add

```
public class Add extends TwoArgExpr {
    public Add( Expression x, Expression y ) {
        super ( x, y,
                (a1, a2) \rightarrow a1 + a2,
                ( p1, p2 ) -> partialEvalAdd( p1, p2 ));
    private static Expression partialEvalAdd( Expression x, Expression y ) {
    @Override
    protected final String getOperator() {
        return "+";
```

Java peculiarities: static imports

- in Java we import a class as
 import java.util.Scanner;
 import java.lang.Math;
- sometimes it is boring/annoying to write class namese.g. Math.PI or Math.sin
- this can be prevented by static importsimport static java.lang.Math.PI;
- allows us to write PI instead of Math.PI
- static import of *all* public methods and attributes from Math:

 import static java.lang.Math.*;



Expression factory

To create an expression we have to write

- hampers readability
- standard Java solution (design pattern): a *factory method*:

```
public static Add add( Expression x, Expression y ) {
   return new Add(x, y);
}
```

Now we can use:

```
Expression e1 = add( mul( const(2), const(3)), var( "x"));
```

typically we introduce a **factory class** containing this kind of methods



Radboud Unive

Finally



