Java OOP
10128

Generics

Pini shlomi

Generics

אפקה
המכללה האקדמית
להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

- Allows writing reusable code that works with various object types.
- Added in Java 5 and is now a core language feature.
- Enables creation of classes, interfaces, and methods with type parameters.
- Benefits
  - Promotes code reusability and flexibility
  - Makes code easier to maintain
  - Helps catch type-related errors at compile-time, simplifying debugging

Errors at compile-time

אפקה
המכללה האקדמית
להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

```
public static void main(String[] args) {
    List list1 = new ArrayList();  // ⚠ Row-Type – need Object Type –
    list1.add("Hello");            // For Supporting previous versions
    list1.add(123);

    for (Object item : list1) {
        String str = (String) item; // ❌ runTime Exception - ClassCastException
        System.out.println(str.toUpperCase());
    }
    List<String> list2 = new ArrayList<>();
    list2.add("Hello");
    list2.add(123); // ❌ compilation Error
}
```

Generic Classes

אפקה
המכללה האקדמית
להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

```
public class Box<T> {
    private T contents;

    public void setContents(T contents) {
        this.contents = contents;
    }

    public T getContents() {
        return contents;
    }
}
public static void main(String[] args) {
  Box<Integer> box1 = new Box<Integer>();
  box1.setContents(42);
  int contents1 = box1.getContents(); // No casting required
  System.out.println("Content is: " + contents1);

  Box<String> box2 = new Box<String>();
  box2.setContents("Hello");
  String contents2 = box2.getContents(); // No casting required
  System.out.println("Content is: " + contents2);
}
```

Console

```
Contents is : 42
Contents is : Hello
```

אפקה
המכללה האקדמית להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

תרגיל 1 : מה יקרה בהרצת הקוד הבא?

```
public class Main {
    public static void main(String[] args) {

        Holder<Number> h1 = new Holder<>(42);
        System.out.println("h1: " + h1.get());

        Holder raw = h1;
        raw = new Holder<>("forty two");
        System.out.println("raw: " + raw.get());

        Holder<Integer> h2 = raw;
        System.out.println("h2: " + h2.get());

        Integer result = h2.get();
        System.out.println("result: " + (result + 1));
    }
}
```

```
class Holder<T> {
    private final T data;

    Holder(T data) {
        this.data = data;
    }

    public T get() {
        return data;
    }
}
```

Wildcard Types

אפקה
המכללה האקדמית
להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

- A wildcard is a type argument that represents an unknown type.
- We use the "?" symbol to specify a wildcard.
- We can use wildcards to create more flexible generic classes and methods.

Wildcard

אפקה
המכללה האקדמית
להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

```
public class Main {
  public static void printList(List<?> list) {
      for (Object item : list) {
          System.out.println("Item: " + item);
      }
      list.add(null); // ✅ Allowed,  throw RunTime error, Why?
      // list.add("new item"); // ❌ Compilation error: cannot add
  }
  public static void main(String[] args) {
      List<String> stringList = Arrays.asList("A", "B", "C");
      List<Integer> intList = Arrays.asList(1, 2, 3);
      printList(stringList);
      printList(intList);
  }
}
```

Arrays.asList create constant size and not allow add/remove, throw UnsupportedOperationException

```
Item: A
Item: B
Item: C
```

```
Item: 1
Item: 2
Item: 3
```

Upper Bounds **<? extends Animal>**

אפקה
המכללה האקדמית
להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

```java
class Animal {
    public void speak() {
        System.out.println("Animal speaks");
    }
}
class Dog extends Animal {
    @Override
    public void speak() {
        System.out.println("Dog barks");
    }
}
class Cat extends Animal {
    @Override
    public void speak() {
        System.out.println("Cat meows");
    }
}
```

```java
public class Main {
    // Accepts a list of Animal or any subclass of Animal
    public static void printAnimalSounds(
                        List<? extends Animal> animals) {
        for (Animal a : animals) {
            a.speak();  // Reading is allowed
        }
        // animals.add(new Dog()); // ❌ Compilation error: can't add
    }
    public static void main(String[] args) {
        List<Dog> dogs = new ArrayList<>();
        dogs.add(new Dog());
        List<Cat> cats = new ArrayList<>();
        cats.add(new Cat());
        printAnimalSounds(dogs);
        printAnimalSounds(cats);
    }
}
```

אפקה
המכללה האקדמית להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

תרגיל 2 : מה יקרה בהרצת הקוד הבא?

```
class Animal {
    void sound() {
      System.out.println("animal");
    }
}

class Cat extends Animal {
    @Override
     void sound() {
            System.out.println("meow");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        List<Cat> cats = new ArrayList<>();
        cats.add(new Cat());
        cats.add(new Cat());
        List<? extends Animal> zoo = cats;
        zoo.add(new Cat());
        System.out.println(zoo.size());
    }
}
```

אפקה
המכללה האקדמית להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

תרגיל 3 : מה יקרה בהרצת הקוד הבא?

```
class Animal {
    void sound() {
        System.out.println("animal");
    }
}

class Cat extends Animal {
    @Override
    void sound() {
            System.out.println("meow");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        List<Cat> cats = new ArrayList<>();
        cats.add(new Cat());
        cats.add(new Cat());
        List<? extends Animal> zoo = cats;
        Animal first = zoo.get(0);
        Cat c = (Cat) first;
        Animal a = first;
        System.out.println(zoo.size());
        System.out.println(c == a);
    }
}
```

Lower Bounds

<? super Dog>

אפקה
המכללה האקדמית
להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

```java
class Animal {
    public void speak() {
        System.out.println("Animal speaks");
    }
}
class Dog extends Animal {
    @Override
    public void speak() {
        System.out.println("Dog barks");
    }
}
class Bulldog extends Dog {
    @Override
    public void speak() {
        System.out.println("Bulldog growls");
    }
}
```

```java
public class Main {
    // Accepts a list of Dog or any superclass of Dog
    public static void addDogs(List<? super Dog> dogList) {
        dogList.add(new Dog());
        dogList.add(new Bulldog());
        // Object obj = dogList.get(0); // ✅ OK, but result is Object
        // Dog d = dogList.get(0); // ❌ Not allowed: unsafe cast
    }
    public static void main(String[] args) {
        List<Animal> animals = new ArrayList<>();
        List<Object> objects = new ArrayList<>();
        addDogs(animals); // Allowed
        addDogs(objects); // Also allowed
    }
}
```

אפקה
המכללה האקדמית להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

תרגיל 4 : מה יקרה בהרצת הקוד הבא?

```
public class Main {
    public static void main(String[] args) {

        List<Animal> animals = new ArrayList<>();
        animals.add(new Animal());

        List<? super Dog> kennel = animals;

        kennel.add(new Dog());
        kennel.add(new Bulldog());

        Dog d = (Dog) kennel.get(0);
        System.out.println(kennel.size());
    }
}
```

```
class Animal { }

class Dog      extends Animal { }

class Bulldog  extends Dog    { }
```

Summary

אפקה
המכללה האקדמית
להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

Wildcard	Example Method	Allowed Action
<? extends T>	printAnimalSounds()	✅ Read ❌ Write
<? super T>	addDogs()	❌ Read as T ✅ Write

<? extends T> can read (Producer)
<? super T> can write (Consumer)

🔁 **PECS** = **P**roducer **E**xtends, **C**onsumer **S**uper

Type Erasure

אפקה
המכללה האקדמית
להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

- Generics in Java use type erasure to ensure backward compatibility with pre-existing code.

- Type erasure removes the type parameter at compile time and replaces it with the upper bound or Object.

```
Box<Integer> box = new Box<Integer>();
box.setContents(42);

// After type erasure:
Box box = new Box();
box.setContents(Integer.valueOf(42));
```

אפקה
המכללה האקדמית
להנדסה בתל אביב
פיני שלומי מהנדס תוכנה 054-4636992

תרגיל 5: איסוף ובקרה במחסן מוצרים

קישור לקובץ התרגיל

קישור ל-starter

אתר להורדת ספריות מ-github.