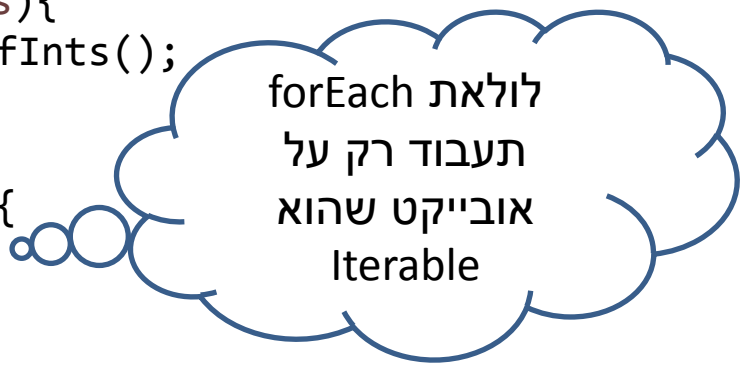
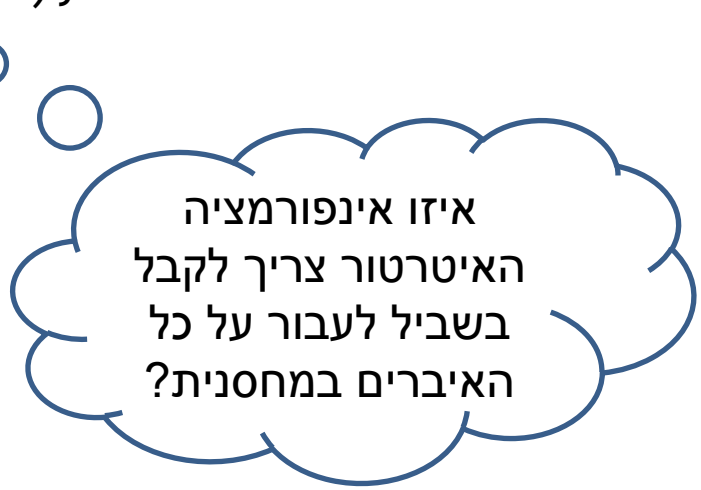
```
public class StackOfInts {
        public static int DEFAULT_STACK_CAPACITY = 10;
        private int[] rep;
        private int count;
        public StackOfInts() {
                count = -1;
                rep = new int[DEFAULT_STACK_CAPACITY];
        }

        public static void main(String[] args){
                StackOfInts sOI = new StackOfInts();
                //some code

                for (Integer currItem : sOI){
                }


                Iterator<Integer> it = sOI.iterator();
                while(it.hasNext()){
                        Integer currItem = it.next();
                }

        }
}
```



```
public class StackOfInts implements Iterable<Integer>{
        public static int DEFAULT_STACK_CAPACITY = 10;
        private int[] rep;
        private int count;
        public StackOfInts() {
                count = -1;
                rep = new int[DEFAULT_STACK_CAPACITY];
        }
        @Override
        public Iterator<Integer> iterator() {
                return new IntStackIt(????);
        }
}
```



```
public class IntStackIt implements Iterator<Integer>{
        private int[] rep;
        private int lastItemIndex;
        private int currIndex;

        public IntStackIt(int[] rep, int lastItemIndex){
                this.rep = rep;
                this.lastItemIndex = lastItemIndex;
                this.currIndex = ????;
        }

        public boolean hasNext() {
                return ????;
        }
        @Override
        public Integer next() {
                return ????;
        }
}
```

```
public class MyList<T>{

        private class Cell {
                private T cont;
                private Cell next;

                public T cont() {
                        return cont;
                }

                public Cell next() {
                        return next;
                }// ...{
        }

        private Cell head;
        // ...
}
```

```
public class MyList<T> implements Iterable<T>{

	private class Cell {
		private T cont;
		private Cell next;
		public T cont() {
			return cont;
		}
		public Cell next() {
			return next;
		}// ...{
	}
	private Cell head;
	// ...

	@Override
	public Iterator<T> iterator() {
		return new MyListIterator<T>(????);
	}
}
```

```
public class MyList<T> implements Iterable<T>{

        //previous code here

        public Iterator<T> iterator() {
                return new MyListIterator(head);
        }


        private class MyListIterator implements Iterator<T>{
                private Cell curr;

                public MyListIterator(Cell head){
                        this.curr = head;
                }

                @Override
                public boolean hasNext() {
                        return ????;
                }
                @Override
                public T next() {
                        return ????;
                }
        }
}
```