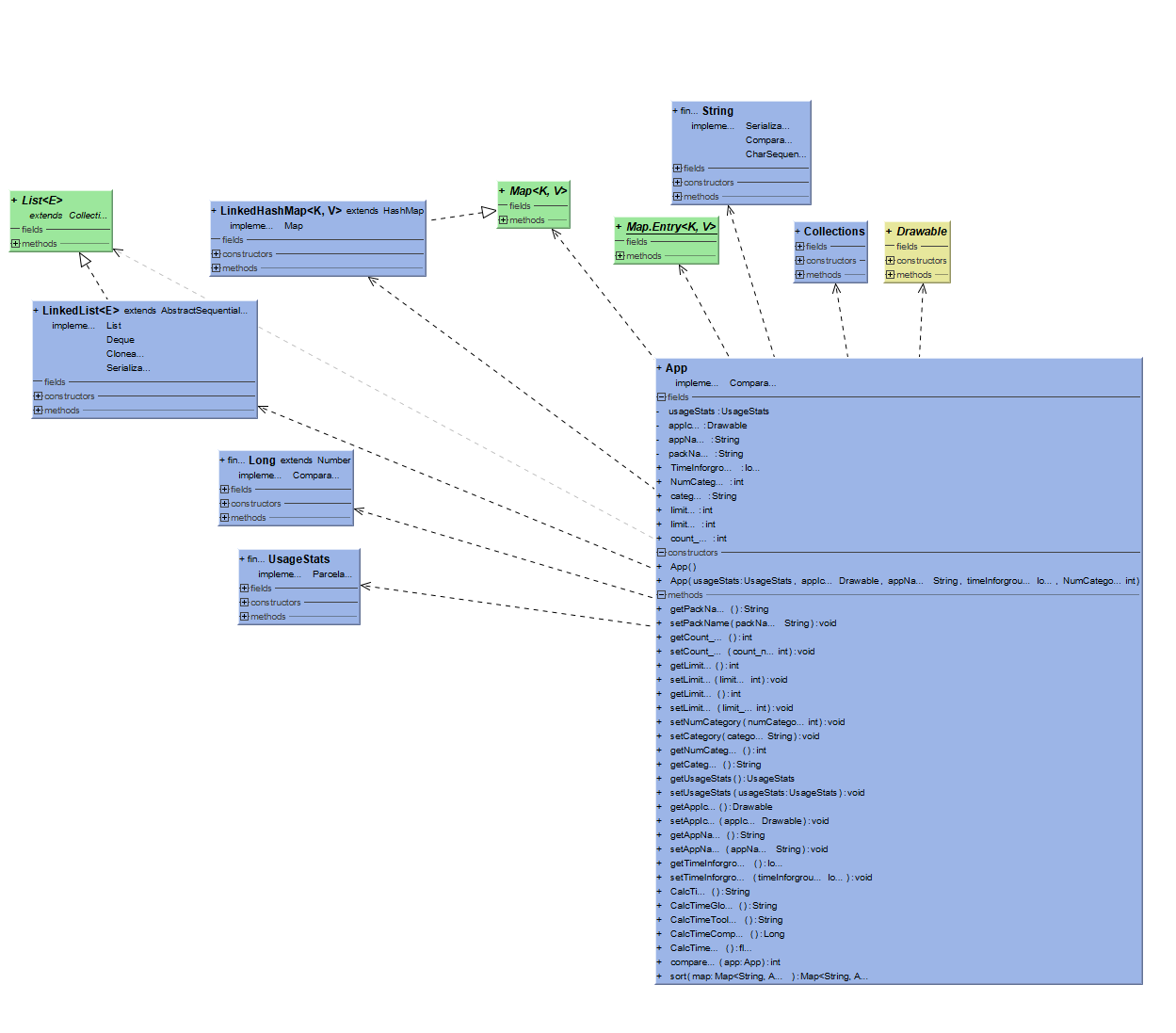
**App( класс содержащий описание приложений)**

****

**Содержит стандартный методы Set и Get**

public String CalcTime () {  
 long minutes = 500, seconds = 500, hours = 500;  
 minutes = (int) ((TimeInforground / (1000 \* 60)) % 60);  
 seconds = (int) (TimeInforground / 1000) % 60;  
 hours = (int) (TimeInforground / (1000 \* 60 \* 60));  
 return hours + "h" + ":" + minutes + "m" +":"+ seconds + "s";  
}  
  
public String CalcTimeGlobal () {  
 long minutes = 500, hours = 500;  
 minutes = (int) ((TimeInforground / (1000 \* 60)) % 60);  
 hours = (int) (TimeInforground / (1000 \* 60 \* 60));  
 return hours + "h " + minutes + "min";  
}  
  
public String CalcTimeToolBar () {  
 long minutes = 500, seconds = 500, hours = 500;  
 minutes = (int) ((TimeInforground / (1000 \* 60)) % 60);  
 seconds = (int) (TimeInforground / 1000) % 60;  
 hours = (int) (TimeInforground / (1000 \* 60 \* 60));  
 seconds = seconds / 60;  
 hours = hours \* 60;  
 minutes+= hours+seconds;  
 return minutes+"";  
  
}  
  
public Long CalcTimeCompare () {  
 long minutes = 500, seconds = 500, hours = 500;  
 minutes = (int) ((TimeInforground / (1000 \* 60)) % 60);  
 seconds = (int) (TimeInforground / 1000) % 60;  
 hours = (int) (TimeInforground / (1000 \* 60 \* 60));  
 seconds = seconds/60;  
 hours= hours \*60;  
 minutes+=hours+seconds;  
 return minutes;  
}  
  
public float CalcTimeBar () {  
 float minutes = 500f, seconds = 500f, hours = 500f;  
 minutes = (float) ((TimeInforground / (1000 \* 60)) % 60);  
 seconds = (float) (TimeInforground / 1000) % 60;  
 hours = (float) (TimeInforground / (1000 \* 60 \* 60));  
 seconds = seconds/60f;  
 hours= hours \*60f;  
 minutes+=hours+seconds;  
 return minutes;  
}

**методы вычисления время использования приложения, отличаются типом возвращаемого значения, в зависимости от требования и надобности.**

@Override  
public int compareTo(@NonNull App app) {  
 if (usageStats == null && app.getUsageStats() != null) {  
 return 1;  
 } else if (app.getUsageStats() == null && usageStats != null) {  
 return -1;  
 } else if (app.getUsageStats() == null && usageStats == null) {  
 return 0;  
 } else {  
 return Long.*compare*(app.TimeInforground,this.TimeInforground);  
 }  
}  
  
public Map<String,App> sort (Map<String,App> map){  
 List<Map.Entry<String, App>> list\_to\_map =  
 new LinkedList<>(map.entrySet());  
 Collections.*sort*(list\_to\_map, new Comparator<Map.Entry<String,App>>() {  
 public int compare(Map.Entry<String, App> o1,  
 Map.Entry<String, App> o2) {  
 return (o1.getValue()).compareTo(o2.getValue());  
 }  
 });  
  
 Map<String, App> sortedMap = new LinkedHashMap<String, App>();  
 for (Map.Entry<String, App> entry : list\_to\_map) {  
 sortedMap.put(entry.getKey(), entry.getValue());  
 }  
  
 return sortedMap;  
}

**Переопределение сортировки мапы по времени использования.**